HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

Fraunhofer
IIS

MASTER THESIS

Mister
Alexander Steinhardt

Consideration of local structures in hierarchical partitioning of integrated circuit netlists modelled by hypergraphs

2013

Fakultät **Mathematik/Naturwissenschaften/Informatik**

MASTER THESIS

Consideration of local structures in hierarchical partitioning of integrated circuit netlists modelled by hypergraphs

Autor:
Alexander Steinhardt

Studiengang:
Diskrete und Computerorientierte Mathematik

Seminargruppe:
ZD10w1

Erstprüfer:
Prof. Dr. rer. nat. Peter Tittmann

Zweitprüfer:
Dipl. Inf. Andy Heinig

Mittweida, Februar 2013

Bibliography

Steinhardt, Alexander: Consideration of local structures in hierarchical partitioning of integrated circuit netlists modelled by hypergraphs, ?? Pages with appendix, 32 Figures, 20 Tables, Hochschule Mittweida (FH), Fakultät Mathematik/Naturwissenschaften/Informatik

Master thesis, 2013

Master thesis

Referat

Die vorliegende Arbeit befasst sich mit der hierarchischen Partitionierung von Hypergraphen, die im Prozess der Chipentwicklung durch Netzlisten einer integrierten Schaltung entstehen. Das Problem der Partitionierung ist dabei NP-schwer, sodass Heuristiken für die entsprechende Partitionierung verwendet werden. Erschwerend kommt hinzu, dass die Hypergraphen meist eine große Ordnung, mehr als 10^5 Knoten, aufweisen. Ziel dieser Arbeit ist es, ein Modell für eine derartige Partitionierung zu erstellen, welches Resultate aus vorangegangenen Arbeiten berücksichtigt und die Platzierung der Elemente des integrierten Schaltkreises im Fokus hat. Des Weiteren wird die Tauglichkeit vorhandener Algorithmen auf die erwähnte Modellierung geprüft, weiter werden vorhandene Algorithmen modifiziert und eigene Algorithmen konzipiert. Dabei wird darauf geachtet, dass die Algorithmen auf Hypergraphen mit großer Ordnung anwendbar sind. Diese Algorithmen werden dazu auf Hypergraphen des Benchmarks "ISPD 05/06" angewandt.

Contents

List of Figures

List of Tables

List of Algorithms

Notation

$\varnothing$	Empty set
$\mathbb{N}$	Set of natural numbers $\{1, 2, 3, \ldots\}$
$\mathbb{N}_0$	$\mathbb{N} \cup \{0\}$
$[n]$	Set of the first n natural numbers $\{1, 2, 3, \ldots, n\}$
$[\text{"condition"}]$	Indicator function that is one if the condition holds; otherwise it's zero.
V	Non empty set
$\lvert V \rvert$	Cardinality of set V
$\mathcal{P}(V)$	Power set of V
$V(H)$	Vertex set of the hypergraph H
$E(H)$	Edge set of the hypergraph H
$r(e)$	Size of an edge e
$\lfloor x \rfloor$	$\max(n \in \mathbb{N} \mid n \leq x)$
$\Rightarrow$	Implication
$\{\}^*$	Multiset
$m_H(v, w)$	Number of edges in H that include v and w.
$\mathrm{idx}(x)$	Index of x, so $\mathrm{idx}(x) := u$ for x_u.
$\mathcal{H}$	Set of all finite hypergraphs.
$\mathcal{IC}$	Set of all finite IC-hypergraphs.
Δ_H	Maximum degree of a hypergraph H.
δ_H	Minimum degree of a hypergraph H.
r_H	Rank of a hypergraph H.
s_H	Anti-rank of a hypergraph H.
$N(v)$	Neighbourhood, all vertices adjacent to $v \in V(H)$.
$N[v]$	Neighbourhood of a vertex $v \in V(H)$ inclusive v.
$N_i(v)$	i-th neighbourhood, all vertices that can be reached from $v \in V(H)$ after at most i steps.
$L_i(v)$	i-th level, the difference $N_i(v) \backslash N_{i-1}(v)$.

1 Introduction

Mathematicians do not study objects,
but relations among objects;...[1]

(Henri Poincaré)

This master thesis participates in research of integrated circuit (IC) design. This design consists of the five essential steps[2] which are specification, coding, synthesis, layout, and manufacturing. Step one to step four can be processed by using the PC and step five brings the IC into physical life. In detail, the step Specification defines the functions which should have the IC. The step Coding describes the behaviour of the IC through a special language like VHDL or Verilog. Next, the step Synthesis transforms the output of the coding with respect to some objective functions. The output of this step is a netlist where the focus of our research lies. This list consists of the information about the elements and the connection between the elements of the IC. The next step, Layout, simulates the placement and wiring of a board with respect to some restrictions like the area, the number of pins of the IC, the temperature development of the IC, that a signal must pass through an element at that time which is given by the clock, and so on. In the last step, Manufacturing, we use the results of step four to wire and place the elements of the IC on a circuit board. Figure ?? shows schematically this IC design process.

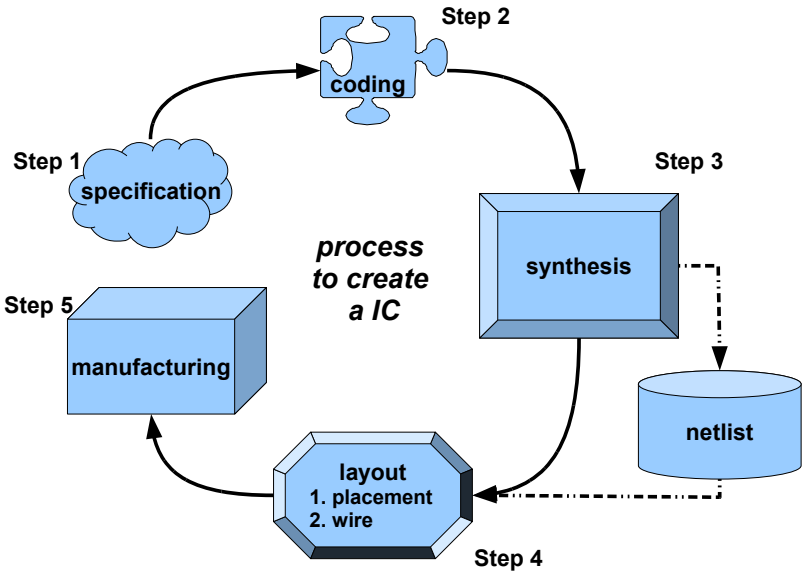


Figure 1.1: Scheme of the process to create an integrated circuit.

[1] See: [?] or [?] in German
[2] See: [?]

The goal of this thesis is to achieve by appropriate partitioning of the netlist a good placement and wiring of the IC. In this context, a good placement and wiring means that the elements consume as few area as possible and a signal reaches its destination at a given time with respect to some restrictions. Further, at step four the placement does not set each element individually because it is too time-consuming. Instead, the top-down strategy is often used. First of all in this strategy the netlist will be partitioned, then the blocks will be placed and wired, and finally the elements will be arranged in the blocks appropriately. Our focus lies on the step to get an appropriate partitioning of a netlist. To solve this partitioning problem we use the hypergraph implied by the netlist and then we partition the hypergraph. Our aim is to partition the hypergraph into $k > 1$ blocks to minimize an objective function. The parameter k should not be too small, otherwise the top-down strategy makes no sense. In the past, the minimization of the cut was a good approach, because the most success was achieved by the placement of the elements due to the fact that they consumed the most space. Therefore, the aim of the wiring was to consume as little space as possible. But in recent years, the size of the elements shrank and a good wiring of the blocks became more important. Because of the lower space consumption of the blocks we have more opportunities for the placement and the wiring should not prevent a good placement. This master thesis shows, in Section ??, that the minimization of the cut is not the best approach to achieve a good placement of the elements of a IC. But at first, we investigate the minimum cut approach from Karypis et al. in this section. For this investigation we need to know what a hypergraph and a hierarchical partitioning is. The field of hypergraphs is introduced in Section ??. Also, we introduce the problem of isomorphism and define the class of integrated circuit hypergraphs (IC-hypergraphs) in this section. This class introduces logic and communication parts to describe the IC. This approach is a consequence of some partial results of the master thesis [**?**] from F. Heinicke. Finally we discuss random walks in this section which will be used in the sections ?? and ??. In Section ??, we define the hierarchical partitioning. There, we briefly introduce the field of categories and introduce two new categories, the hierarchical category and a subcategory of it, the SA-category. Also, we show that the hierarchical partitioning of hypergraphs belongs to the hierarchical category or to the SA-category with respect to some restrictions. In the final section, Section ??, we summarize the results and provide an outlook for further studies.

2 Hypergraphs

A hypergraph $(e_1, e_2, \ldots, e_m)$ of a set V according to [**?**] or [**?**] is a collection of subsets of a given set V which satisfies:

(i) $e_i \neq \emptyset$ for all $i = 1, \ldots, m$

(ii) $\bigcup_{i \in [m]} e_i = V$

We use a more general approach[1] and define a *hypergraph* $H = (V, E)$ as an ordered pair (V, E) comprising a set V of *vertices* together with a multiset E of *edges* or *hyperedges* of nonempty subsets of V. So we have:

(i) $e \neq \emptyset$ for all $e \in E$

(ii) $\bigcup_{e \in E} e \subseteq V$

We call a hypergraph (V', E') *subhypergraph* of H if $V' \subseteq V$ and $E' \subseteq E$. Next, let $H[V'] = (V', \{e \in E(H) \mid e \subseteq V'\}^*)$ be the by $V' \subseteq V$ induced subhypergraph of H. A vertex which belongs to no edge is called *isolated*. If two edges $e_j, e_k \in E(H)$ with $j, k \in [E(H)]$ and $j \neq k$ are equal then we call them *parallel*. Another special edge is a *loop* which is an edge with cardinality one. A hypergraph is called *simple* if it contains no parallel edges and no loops. Furthermore, a hypergraph is called *multigraph* if all edges have a cardinality of at most two, and a multigraph is called *graph* if it has a set instead of a multiset of edges. Therefore, a hypergraph is a generalisation of a multigraph in such a way that the restriction to the cardinality does not have to be satisfied. Further, a multigraph is a generalisation of a graph in such a way that we have a multiset instead of a set of edges. A graph is called *bipartite* if the vertex set V can be splitted into two disjoint sets V_1 and V_2 so that every edge connects a vertex of V_1 with exactly one in V_2.

Example 2.1:
$V = [8]$
$E = \{e_1 = \{1\},\ e_2 = \{1, 2\},\ e_3 = \{2, 3\},\ e_4 = \{2, 3\},\ e_5 = \{2, 5, 6\},\ e_6 = \{5, 6, 7, 8\}\}^*$
$H' = (V, E)$

[1] We do not use the definition of [**?**] or [**?**] because we want a real generalisation of graphs or rather multigraphs.

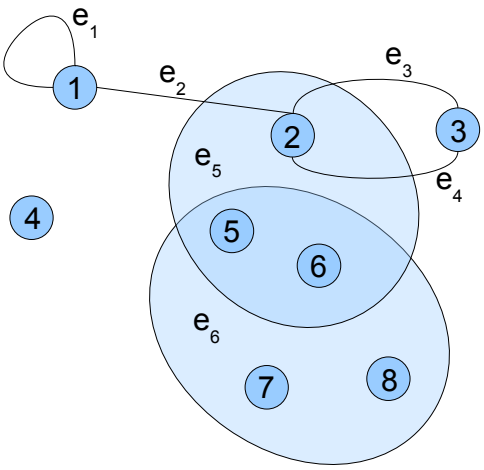


Figure 2.1: Hypergraph H'

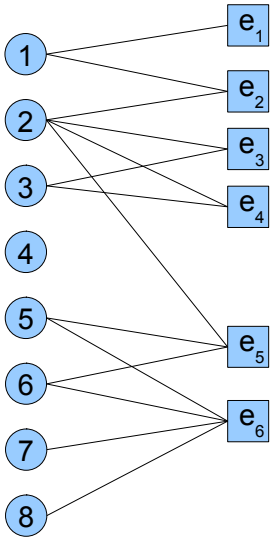


Figure 2.2: Bipartite representation of H'.

The illustration of H' in Figure ?? uses lines for edges which have a cardinality of at most two. The other edges are drawn as ellipses[2]. Figure ?? is the bipartite representation of H', where the *bipartite representation of a hypergraph* $H = (V, E)$, written as $\text{bipart}(H) = (V \cup V', E')$, is a graph that contains V and new vertices $e_1, \ldots, e_m \in V'$ that represent the edges of H. In figure ?? this new vertices are drawn as squares. The edges of this bipartite graph consist of all pairs of vertices $v \in V$ and $e \in V'$ where v belongs to e in H. In other words, we have $e \in E(\text{bipart}(H))$ if and only if e contains $v \in V$ and $w \in V'$ where w is a representative of $f \in E$ that satisfies $v \in f$. The edges of this graph are drawn as lines. The following algorithm can be used to create the bipartite representation.

Algorithmus 1: bipart

Input: A finite hypergraph H'.

Output: The bipartite representation of H'.

1: $V = V(H'), E = \emptyset$
2: **for** $e \in E(H')$ **do**
3: Create a new vertex v which represent e and insert them to V.
4: **for** $w \in e$ **do**
5: $E = E \cup \{v, w\}$
6: **end for**
7: **end for**
8: **return** (V, E)

[2] In later figures we also use other shapes.

It is easy to see that the runtime is

$$\mathcal{O}\big(|E(H)| \cdot \max\left(\{|e| \mid e \in E\}\right)\big) \subseteq \mathcal{O}(|E(H)| \cdot |V(H)|). \tag{2.1}$$

Remark:
The output of bipart is a simple bipartite graph.

The *order* of $H = (V, E)$, written as $ord(H) = |V|$, is the cardinality of the vertex set. If $ord(H) \in \mathbb{N}$ then we call H *finite* otherwise *infinite*. We restrict our studies to finite hypergraphs. In Example ?? the order of H' is eight. We call a vertex $v \in V$ *incident* to an edge $e \in E$ if $v \in e$ is satisfied. The *degree* $deg(v)$ of a vertex $v \in V$ is the number of edges which are incident to v. The number of vertices that are incident to an edge $e \in E$ is defined by $r(e) = |e|$. For instance we have $deg(2) = 4$ and $r(e_5) = 3$ in Example ??. Further, we define the maximum degree by $\Delta_H = \max_{v \in V} deg(v)$ and the minimum degree by $\delta_H = \min_{v \in V} deg(v)$. In Example ?? the maximum degree is three and the minimum degree is zero. Next the *rank* of H is $r_H = \max_{e \in E} r(e)$ and the *anti-rank* of H is $s_H = \min_{e \in E} r(e)$. If $r_H = s_H$ we say that H is a *uniform* hypergraph. In Example ?? the rank is four and the anti-rank is one. The *dual* hypergraph H^* of $H = (V, E)$ is a hypergraph with vertex set $E' = \{e_1, \ldots, e_m\}$ and a multiset of edges $\{X_1, \ldots, X_{|V|}\}^*$ where every X_j is defined by $X_j := \{e_i \in E' : v_j \in V,\ v_j \text{ incident to } e_i \text{ in } H\}$ for all $j \in [|V|]$.

Remark:
We obtain the dual hypergraph H^* of $H = (V, E)$ simply from bipartite representation $\mathrm{bipart}(H) = (V \cup V', E')$ of H. We only need to interpret the elements of V' as vertices and the elements of V as edges in $\mathrm{bipart}(H)$.

A *path* $v_1 f_1 v_2 \cdots v_k f_k v_{k+1}$ of length $k \in \mathbb{N}$ between two different vertices v_1, v_{k+1} is an alternating sequence of pairwise different vertices $v_i \in V(H)$ with $i \in [k+1]$ and pairwise different edges $f_j \in E(H)$ with $v_j \in f_j$ and $v_{j+1} \in f_j$ for all $j \in [k]$.

Remark:
The edges in a path in a simple graph are redundant, so that a path between to vertices v_1, v_k could be written as a sequence $v_1 v_2 \ldots v_{k-1} v_k$ of vertices.

A *cycle* is a path between two vertices v_1, v_k with the exception that $v_1 = v_k$ holds. A hypergraph is called *forest* if it contains no cycle. A path between to vertices is called *shortest path* if there is no other path between this two vertices with fewer edges. The *distance* $d(v, w)$ between two vertices $v, w \in V(H)$ is defined by the number of edges of a shortest path between v and w. If there is not such a shortest path between two vertices v, w then we define $d(v, w) := \infty$ and further we define $d(v, v) := 0$. In Example ?? the sequence $1e_2 2e_5 5e_6 8$ is a shortest path between 1 and 8 with a distance of $d(1, 8) = 3$. The *eccentricity* $ecc(v)$ of a vertex v is the maximal distance of v to any other vertex in the set of vertices. Formally it means $ecc(v) = \max_{u \in V(H)} \{d(v, u)\}$. A hypergraph is *connected* if there is a path between any pair of vertices of

this hypergraph. Further a maximal connected subhypergraph of H is called *component of* H. A cycle free hypergraph is called *forest* and a connected forest is called *tree*. The *neighbourhood* of a vertex $v \in V$, written as $N(v)$, is the set of all vertices $w \in V$ with $d(v,w) = 1$. We write $N[v]$ if we include a vertex v to its neighbourhood so $N[v] := \{w \in V \mid d(v,w) \leq 1\}$. If we have a subset $W \subseteq V$ of V, so we define the neighbourhood of W as the union of the neighbourhoods of each of the vertices in W without W itself. Formally it means $N(W) = \bigcup_{v \in W} N(v) \backslash W$. We also write $N[W]$ if we include W to $N(W)$. The *i-th neighbourhood* is $N_i(v) := \{w \in V \mid d(v,w) \leq i\}$ for any $i \in \{0, \ldots, ecc(v)\}$. The *i-th level* $L_i(v)$ of a vertex v is the set of all vertices that can be reached from v by a shortest path of length i. So, we define $L_i(v) := \{w \in V \mid d(v,w) = i\}$ for any $i \in \{0, \ldots, ecc(v)\}$. Furthermore, we assume w.l.o.g. that $V = [n]$ with $n \in \mathbb{N}$.

Remark:
If $V = [n]$ then hypergraphs without isolated vertices are characterized by their edge set.

A *partition* $P = \{B_1, \ldots, B_k\}$ of the vertex set V into $k \in \mathbb{N}$ blocks is a set of k disjoint nonempty subsets of V that cover all of V. Formally it means that $B_i \cap B_j = \emptyset$ for all $i, j \in [k]$ with $i \neq j$ and $\bigcup_{i=1}^{k} B_i = V$ holds. A *cut* of a hypergraph H with respect to a partition $P = \{B_1, \ldots, B_k\}$ is a set of edges whose removal will disconnect the blocks $B_1, \ldots, B_k$. A *minimum cut* of H with respect to a partition $P = \{B_1, \ldots, B_k\}$ is the smallest cut of H with respect to P that will pairwise disconnect the blocks $B_1, \ldots, B_k$. If $H = (V, E)$ is partitioned into k nonempty blocks $B_1, \ldots, B_k$ and the partition is dented by $P = \{B_1, \ldots, B_k\}$ then the *block connectivity graph of* H, written as $\mathrm{BC}(H, P) = ([k], F)$, is the graph whose vertices are the block numbers and the edge set F is the set of all sets $\{i, j\}$ with $i, j \in [k]$ and $i \neq j$ if there is an edge in E between a vertex with block number i and a vertex with block number j.[3] If the block connectivity graph of H is a forest or a tree then we call this graph *block connectivity forest of* H or *block connectivity tree of* H. Algorithm ?? presents the steps to get the block connectivity graph.

[3] Observe that a hyperedge can induce more than one edge in the block connectivity graph.

Algorithmus 2: getBC

Input: A finite partitioned hypergraph $H' = (V', E')$ with partition P.

Output: The block connectivity hypergraph $BC(H', P) = BC$.

```
for B ∈ P do
    Insert a vertex B in BC.
end for
for v ∈ V' do
    B = block(v)
    for w ∈ N(v) do
        if w ∉ {x ∈ V' | block(x) = B} and {B, block(w)} ∉ E(BC) then
            Insert {B, block(w)} into E(BC).
        end if
    end for
end for
return BC
```

The runtime of Algorithm ?? is $\mathcal{O}(|V(H')|^2)$ where the number of neighbours of each vertex was estimated by $|V(H')|$.

Example 2.2:

This example shows a hypergraph that is partitioned into three blocks, which are coloured red, green, and blue.

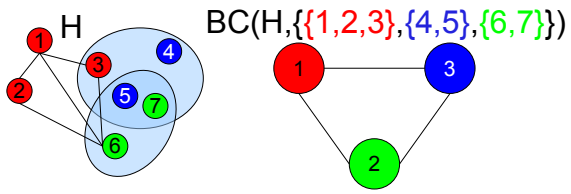


Figure 2.3: A partitioned hypergraph H and its block connectivity graph.

We see that, if the edge which includes the vertices $5, 6,$ and 7 would not exist then the block connectivity graph would not change.

From another point of view, a hypergraph can be understood as a structure of relations. Because an edge with $k \in [|V(H)|]$ elements can be taken as a k-ary relation and vice versa.

2.1 Hypergraph operations

In this section we briefly discuss some basic hypergraph operations. An *operation on a hypergraph* $H = (V, E)$ is a map from H to another hypergraph H'. This map is called a *morphism* into the set of hypergraphs $\mathcal{H}$. The first morphism we define is the binary morphism

$$+ : \mathcal{H} \times M \to \mathcal{H} : +(H, \{v\}) \mapsto (V \cup \{v\}, E)$$

where M is a set. We also write $H + \{v\}$ for $+(H, \{v\})$, and this morphism acts on H so that v will be added to the vertex set of H. Note that it is also possible to add a set of vertices to H. Next, we define the binary morphism

$$\tilde{+}_H : E \times M \to \mathcal{H} : \tilde{+}(e, v) \mapsto (V \cup \{v\}, (E \backslash e) \cup (e \cup \{v\})).$$

We also write $e\tilde{+}_H \{v\}$, or $e\tilde{+} \{v\}$ if it is clear which hypergraph is meant, for $\tilde{+}_H(e, \{v\})$, and this morphism acts on the edge $e \in E$ so that v will be added to e. Note that it is also possible to add a set of vertices to an edge e of H. The third morphism we define is the binary morphism

$$\hat{+} : \mathcal{H} \times M \to \mathcal{H} : \hat{+}(H, \{e\}) \mapsto (V \cup e, E \cup \{e\})$$

where M is a set. We also write $H\hat{+} \{e\}$ for $\hat{+}(H, \{e\})$, and this morphism acts on H so that e will be added to E and the vertices of e, which are not in V, will be added to V. Note that it is also possible to add a set of edges to H. On the other hand, it is also possible to delete an edge from H. It is defined by the binary morphism

$$\hat{-} : \mathcal{H} \times M \to \mathcal{H} : \hat{-}(H, \{e\}) \mapsto (V, E \backslash \{e\})$$

where M is a set. We also write $H\hat{-} \{e\}$ for $\hat{-}(H, \{e\})$. Note that it is also possible to delete a set of edges from H. Next, we define the binary morphism

$$\begin{aligned} &- : \mathcal{H} \times M \to \mathcal{H} : \\ &-(H, \{v\}) \mapsto (V \backslash \{v\}, (E \backslash \{f \in E \mid v \in f\}) \cup \{f \backslash \{v\} \mid f \in E, v \in f, |f \backslash \{v\}| > 0\}) \end{aligned}$$

where M is a set. We also write $H - \{v\}$ for $-(H, \{v\})$, and this morphism acts on H so that if $v \in V$ then v will be deleted from all edges of H that contain v and from V. Note that it is also possible to delete a set of vertices from H. To delete a vertex from an edge we define the binary morphism

$$\tilde{-}_H : E \times M \to \mathcal{H} : \tilde{-}(e, \{v\}) \mapsto (V, (E \backslash e) \cup (e \backslash \{v\})).$$

We also write $e\tilde{-}_H \{v\}$, or $e\tilde{-} \{v\}$ if it is clear which hypergraph is meant, for $\tilde{-}_H(e, \{v\})$. Note that it is also possible to delete a set of vertices from $e \in E$. The next binary morphism $/$ is called *(edge) contraction*. It is defined by

$$\begin{aligned} &/ : \mathcal{H} \times M \to \mathcal{H} : \\ &/(H, \{e\}) \mapsto (M', (E \backslash M'') \cup \{f \backslash (f \cap e) \cup \{w\} \mid f \in M'', |f \backslash (f \cap e) \cup \{w\}| > 1\}) \end{aligned}$$

where M is a set, $e = \{v_1, \ldots, v_k\}$, $w = v_1 \cdots v_k$, $M' = (V \backslash e) \cup \{w\}$, and $M'' = \{f \in E \mid f \cap e \neq \emptyset\}$. We also write $H/\{e\}$ for $/(H, \{e\})$. Note that it is also possible to contract a set of edges in H.

Lemma 2.3:
The morphism $/$ of two vertex disjoint edges is commutative.

Proof: Let $H = (V, E)$ be a hypergraph and $e = \{v_1, \ldots, v_k\}, f = \{v'_1, \ldots, v'_{k'}\} \in E$ with $e \cap f = \emptyset$. Further, let $w = v_1 \cdots v_k$ and $w' = v'_1 \cdots v'_{k'}$. We show that $H/e/f = H/f/e$. We have

$$\begin{aligned}(((V \backslash e) \cup \{w\}) \backslash f) \cup \{w'\} &\overset{w \notin f}{=} (V \backslash (e \cup f)) \cup (\{w\} \cup \{w'\}) \\ &\overset{w' \notin e}{=} (((V \backslash f) \cup \{w'\}) \backslash e) \cup \{w\}),\end{aligned}$$

and from this, the commutativity with respect to the vertex set of H follows. We also have

$$\begin{aligned}(((E \backslash M_r) \cup M'_r) \backslash N_s) \cup N'_s &\overset{M'_r \cap N_s = \emptyset}{=} (E \backslash (M_r \cup N_s)) \cup (M'_r \cup N'_s) \\ &\overset{N'_s \cap M_r = \emptyset}{=} (((E \backslash N_s) \cup N'_s) \backslash M_r) \cup M'_r\end{aligned}$$

with $M_r = \{r \in E \mid r \cap e \neq \emptyset\}$, $N_s = \{s \in E \mid s \cap f \neq \emptyset\}$,
$M'_r = \{(r \backslash (r \cap e)) \cup \{w\} \mid r \in E, r \cap e \neq \emptyset, |(r \backslash (r \cap e)) \cup \{w\}| > 1\}$, and
$N'_s = \{(s \backslash (s \cap f)) \cup \{w'\} \mid s \in E, s \cap f \neq \emptyset, |(s \backslash (s \cap f)) \cup \{w'\}| > 1\}$ where $e \cap f = \emptyset$ implies $M'_r \cap N_s = \emptyset$ and $N'_s \cap M_r = \emptyset$. From this, the commutativity with respect to the edge set of H follows. □

The last operation we consider is the binary morphism

$$\begin{aligned}&* : \mathcal{H} \times M \to \mathcal{H} : \\ &*(H', \{u, v\}) \mapsto ((V' \backslash \{u, v\}) \cup \{uv\}, (E' \backslash M') \cup \{f \backslash \{u, v\} \cup \{uv\} \mid f \in M'\})\end{aligned}$$

that is called *fusion* where $H' = (V', E')$, $M' = \{f \in E \mid f \cap \{u, v\} \neq \emptyset\}$, and M is a subset of the vertex set of H'. We also write $H_{\{uv\}}$ for $*(H, \{uv\})$. Note that it is also possible to fuse a set of more than two vertices from H, and we can image the fusion as a contraction of a pseudo-edge that does not exist in E. Figure ?? illustrates three hypergraph operations, namely the deletion of a vertex, the contraction of an edge, and the fusion of two vertices.

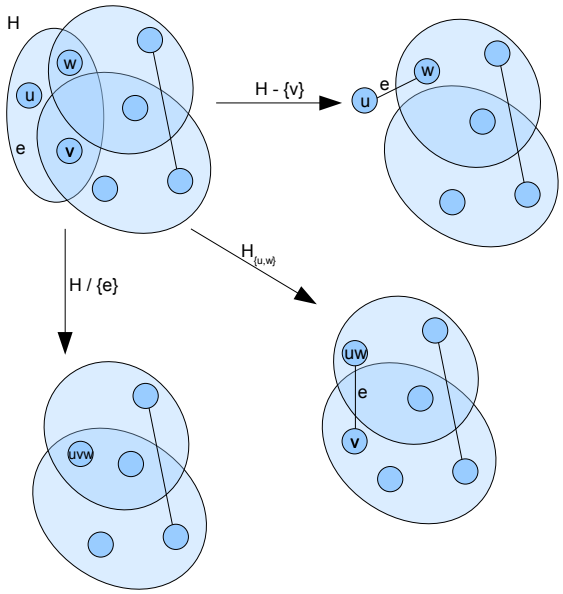


Figure 2.4: Three hypergraph operations.

2.2 Isomorphism

Let $H = (V, E)$ and $H' = (V', E')$ be hypergraphs with $m \in \mathbb{N}$ edges. This two graphs are isomorphic to each other if they have the same structure. So, we can morph each of them into the other one. We introduce two definitions that we can use to decide which two hypergraphs are isomorph. For the first definition we have to order the edges in an arbitrary way. Let $F = (e_1, \ldots, e_m)$ be the collection of the edges of E that are ordered in an arbitrary fashion, and let $F' = (e'_1, \ldots, e'_m)$ be the collection of the edges of E' that are ordered in an arbitrary fashion, too. So, we call H *isomorphic* to H', written as $H \simeq H'$, if there exists a bijection $\varphi : V \to V'$ and a permutation π of $[m]$ so that $\varphi(e_i) = e'_{\pi(i)}$ for all $i \in [m]$ where $\varphi(\{v_1, \ldots, v_k\}) := \{\varphi(v_1), \ldots, \varphi(v_k)\}$ for $v_1, \ldots, v_k \in V$[4]. This bijection φ is called *isomorphism* between H and H'. The two hypergraphs are *equal*, written as $H = H'$, if $H \simeq H'$ is satisfied and φ is the identity permutation.

Example 2.4:
Example of two isomorphic graphs G and G'. We have:

[4] See: [**?**] page 1

$V = V' = \{1, 2, 3, 4\}$
$E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$
$E' = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$
$G = (V, E)$, $G' = (V', E')$
$G = G$
$G \simeq G'$ with $\varphi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$
or also with $\varphi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 3 & 4 \end{pmatrix}$

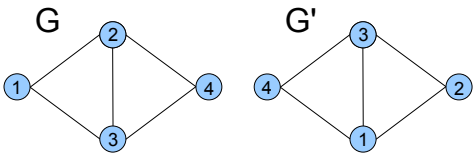


Figure 2.5: Two Graphs G and G' which are isomorphic.

The second definition is a more combinatorial one. Let $G = (X, E)$ and $G' = (X', E')$ be multigraphs. We call a bijection $\varphi : X \to X'$ an isomorphism between G and G' if and only if $m_G(x, y) = m_{G'}(\varphi(x), \varphi(y))$ holds for all $x, y \in X$, where $m_G(x, y)$ is the number of edges which join x and y in G[5]. Furthermore, $M_H(v, w)$ is the pair that comprises the number of edges between v and w of cardinality less or equal two and the number of edges between v and w of cardinality greater than two. So, we call a bijection $\varphi : V \to V'$ an isomorphism between H and H' if and only if $M_H(v, w) = M_{H'}(\varphi(v), \varphi(w))$ holds for all $v, w \in V$.

Example 2.5:
Example of two isomorphic graphs H and H' and one hypergraph H'' that is not isomorphic to H and H'. We have:

$V = V' = V'' = \{1, 2, 3, 4\}$
$E = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}\}^*$
$E' = \{\{1, 3, 4\}, \{1, 3\}, \{2, 3\}\}^*$
$E'' = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$
$H = (V, E)$, $H' = (V', E')$,
$H'' = (V'', E'')$
We have $H \simeq H'$
with $\varphi = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 1 & 2 \end{pmatrix}$
but $H \not\simeq H''$ and $H' \not\simeq H''$.
Because:

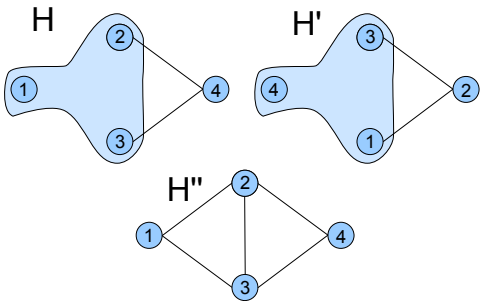


Figure 2.6: Two isomorph hypergraphs H and H' and one, H'', that is not isomorph to H or H'.

(v, w)	$(1, 2)$	$(1, 3)$	$(2, 3)$	$(2, 4)$	$(3, 4)$
$M_H(v, w)$	$(0, 1)$	$(0, 1)$	$(0, 1)$	$(1, 0)$	$(1, 0)$
$M_{H'}(\varphi(v), \varphi(w))$	$(0, 1)$	$(0, 1)$	$(0, 1)$	$(1, 0)$	$(1, 0)$
$M_{H''}(v, w)$	$(1, 0)$	$(1, 0)$	$(1, 0)$	$(1, 0)$	$(1, 0)$

[5] See [?] page 3, or [?] page 25 for graphs.

Remark:
If we consider only hypergraphs with the same number of edges then $m_H(x, y)$ is sufficient for the definition of an isomorphism between this hypergraphs.

Example ?? shows that if an bijection φ is given with respect to two hypergraphs H and H' then it is possible to check if φ is an isomorphism in polynomial time. We only need to build a table as we see in Example ??. But it is still an open problem to get an isomorphism between two given (hyper)graphs in polynomial time[6]. Besides, no-one knows the complexity class of the isomorphism problem.

2.3 Class of integrated circuit hypergraphs

A netlist which represents an IC is designed to solve a problem or several problems. So we could partition the IC in two kinds of blocks: the *logic parts* which are used to solve a problem or deliver a special functionality that is used for a solution of a problem, and the *communication parts* which are used to handle the communication - the decision of which signal should be sent to which part - between the logic parts and themselves. Further, an *IC-hypergraph* from a netlist N is defined by the hypergraph whose vertices are the elements of N and whose edges are the wires of N.

Example 2.6:
This example shows a circuit of a four bit counter[7] and its IC-hypergraph FB with the following data:

Name	Number of vertices	Number of edges	δ	Δ	Average degree	s	r	Average number of vertices in an edge
FB	46	45	1	4	≈ 2.7174	2	7	≈ 2.7778

Figure ?? shows FB.

[6] See: [**?**] page 285
[7] See: [**?**]

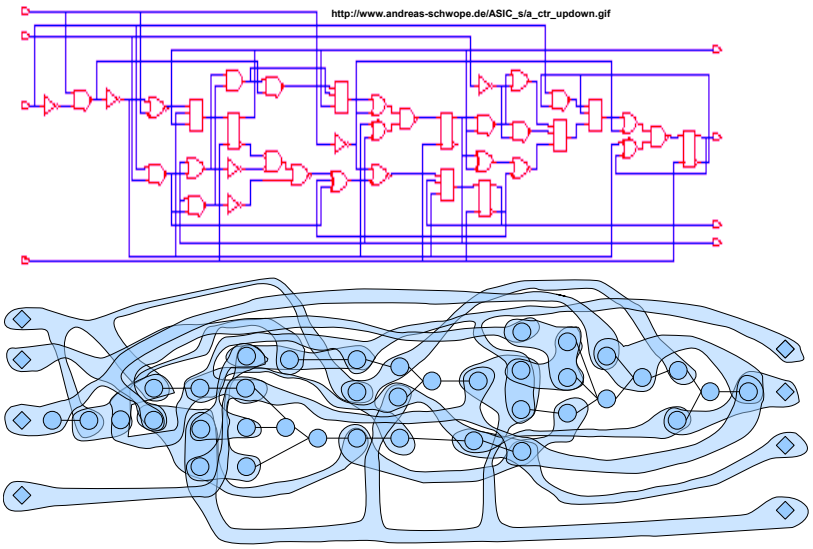


Figure 2.7: Four bit counter and its hypergraph.

If we partition the IC-hypergraph into k blocks with respect to its logic and communication parts then we get k_1 logic parts $L_1, \ldots, L_{k_1} \subseteq V(H)$ and k_2 communication parts $C_1, \ldots, C_{k_2} \subseteq V$ with $k = k_1 + k_2$ and $V = \bigcup_{i=1}^{k_1} L_i \cup \bigcup_{i=1}^{k_2} C_i$. Figure ?? shows a schema of the parts of an IC-hypergraph with k_1 logic and k_2 communication parts.

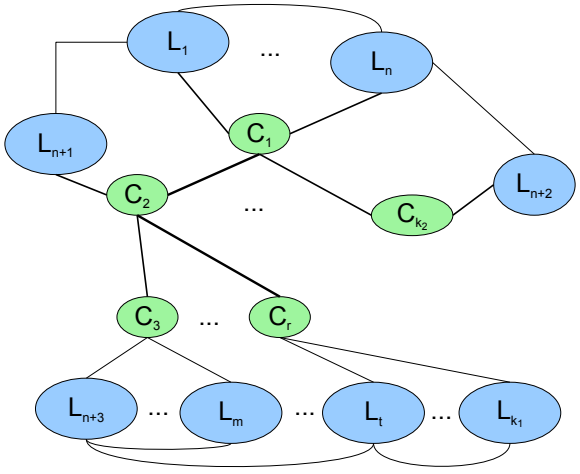


Figure 2.8: Schema of logic and connected parts of a IC-hypergraph H

The thickness of the lines represents the number of edges between the parts. We assume that there are few edges between the logic parts, for example those which occur from the clock or the reset. On the other hand, there are more edges between the communication parts. We define an IC-hypergraph class $\mathcal{IC}_{\varepsilon,\varepsilon'}$ of non-isomorphic[8] IC-hypergraphs with respect to $\varepsilon, \varepsilon' \in \mathbb{N}_0$ that determines which average degree and which average edge cardinality of a hypergraph will be excepted. So, each element of the IC-hypergraph is a representative of the class of hypergraphs that are isomorph to this element. Further, IC-hypergraphs belong to $\mathcal{IC}_{\varepsilon,\varepsilon'}$ if they satisfy the

[8] Discussed at Section ??

following conditions:

(i) $|E(H)| = \mathcal{O}(|V(H)|)$

(ii) $\frac{1}{|V(H)|} \sum_{v \in V(H)} deg(v) \in [2, 3 + \varepsilon)$

(iii) $\frac{1}{|E(H)|} \sum_{e \in E(H)} |e| \in [2, 3 + \varepsilon')$

We define $\mathcal{IC} := \bigcup_{\varepsilon, \varepsilon' \in \mathbb{N}_0} \mathcal{IC}_{\varepsilon, \varepsilon'}$. The four bit counter of Example **??** has an average degree of 2.7174 and an average edge size of 2.7778. So this IC-hypergraph belongs to $\mathcal{IC}_{0,0}$.

2.4 Random walk

A *random walk* on a hypergraph is a special markov chain, so we introduce markov chains at first.

2.4.1 Markov chains[9]

A *stochastic process* $\mathbf{X} = \{X_t \mid t \in T\}$ on the probability space $(\Omega, \mathcal{A}, Pr)$ with time $T \subseteq \mathbb{R}$ is a collection $\{X_t \mid t \in T\}$ of random variables $X_t(\omega) := X(\omega, t) : \Omega \times T \to S$ with values in the *set of states* S. If S is countably we call $\mathbf{X}$ *discrete state process*, and if S is finite we call $\mathbf{X}$ *finite state process*. If T is countably we call $\mathbf{X}$ *discrete time process*. If S and T are countably then $\mathbf{X}$ is a *discrete process*. A *Markov chain* is a discrete stochastic process $\mathbf{X}$ with $T = \mathbb{N}_0$ which satisfies the so called Markov property or memoryless property

$$Pr\left(X_t = j \mid X_{t-1} = i, X_{t-2} = j_{t-2}, \ldots, X_0 = j_0\right) = Pr\left(X_t = j \mid X_{t-1} = i\right)$$

for $i, j, j_{t-2}, \ldots, j_0 \in S$.

Remark:
A *Markov process* is a stochastic process with an uncountably set of states or with uncountably random variables that satisfy the Markov property.

We call $P_{i,j} := Pr\left(X_t = j \mid X_{t-1} = i\right)$ the *transition probability* from $i \in S$ to $j \in S$ which is the probability that the process moves from state i to state j. We call a Markov chain homogeneous if the transition probability is independent of the time for all pairs of states. We restrict our studies to this type of Markov chains. We collect the transition probabilities between every pair of states in the *transition matrix* $P := (P_{i,j})_{i,j \in S}$. The transition matrix is a stochastic matrix which means that $\sum_{j \in S} P_{i,j} = 1$ for all $i \in S$ holds. Furthermore, the probability to be at state i at

[9] See: [**?**]

time t is $p_i(t)$. So we have

$$p_i(t) = \sum_{j \in S} p_j(t-1) P_{j,i}$$

for all $i \in S$, or in matrix notation

$$p(t) = p(t-1)P$$

where $p(t) = (p_i(t))_{i \in S}$ is the distribution of the states at time t. The *m-step transition probability* $P^m_{i,j} := Pr\left(X_{t+m} = j \mid X_t = i\right)$ is the probability that the chain moves from state i to state j in exactly m steps. We have

$$P^m_{i,j} = \sum_{k \in S} P_{i,k} P^{m-1}_{k,j}$$

for all $i, j \in S$, and in matrix notation

$$P^{(m)} = P \cdot P^{m-1}$$

where $P^{(m)} := \left(P^m_{i,j}\right)_{i,j \in S}$ is the matrix of all m-step transition probabilities. So, by induction on m we have

$$P^{(m)} = P^m.$$

Further, we have $p(t+m) = p(t)P^m$. A *stationary distribution* π of a Markov chain is a probability distribution of the states which satisfies $\pi = \pi P$. States $i, j \in S$ are *connected* with each other if there are $k, l \in \mathbb{N}$ with $P^k_{i,j} > 0$ and $P^l_{j,i} > 0$. The relation induced by the connectivity is an equivalence relation that implies a classification of the states which are connected with each other. We call the classes which appear by this classification *communication classes*. A Markov chain is called *irreducible* if all states belong to one communication class; otherwise it is called *reducible*. A Markov chain could be represented by a directed graph[10] whose vertices are the states, and whose arcs are the transitions with positive transition probability.

Lemma 2.7:
A finite Markov chain is irreducible if and only if its graph representation is a strongly connected graph[11].

Proof: By definition, a Markov chain is irreducible if and only if all states belong to one communication class. That means that every state could be reached from all other states. It follows, that the directed graph from the graph representation of this Markov chain is strongly connected. □

The probability $r^t_{i,j} := Pr(X_t = j \text{ and for all } 1 \leq s < t \text{ is } X_s \neq j \mid X_0 = i)$ for $i, j \in S$ and $t \in T$ is the probability to reach j from i after exactly t steps for the first time. We call a state i *recurrent* if $\sum_{t \geq 1} r^t_{i,i} = 1$ and *transient* if $\sum_{t \geq 1} r^t_{i,i} < 1$ holds. A Markov chain is called recurrent if all states are recurrent. If we start at $i \in S$ then we denote the *expected time* to first reach $j \in S$

[10] See: [**?**]
[11] See: [**?**] page 128

by $h_{i,j} := \sum_{t \geq 1} t \cdot r_{i,j}^t$. We call a recurrent state i *positive recurrent* if $h_{i,i} < \infty$; otherwise we call it *null recurrent*.

Lemma 2.8:
The following statements hold for finite Markov chains:

(i) At least one state is recurrent.

(ii) All recurrent states are positive recurrent.

A state i in a discrete Markov chain is *periodic* if there exists an integer $d > 1$ so that $Pr(X_{t+s} = i \mid X_t = i) = 0$ unless d divide s; otherwise i is *aperiodic*. A discrete Markov chain is periodic if all states are periodic; otherwise it is aperiodic. An aperiodic and positive recurrent state is an *ergodic* state. A Markov chain is ergodic if all states are ergodic.

Corollary 2.9:
A finite Markov chain is ergodic if and only if the chain is irreducible and aperiodic.

Lemma 2.10:
For any irreducible and ergodic Markov chain and for any $i \in S$ we have

$$\lim_{t \to \infty} P_{i,i}^t = \frac{1}{h_{i,i}} < \infty.$$

Proof: See [**?**] page 377 et sequens. □

Theorem 2.11:
Any finite, irreducible and ergodic Markov chain with states $S = [n]$ and $n \in \mathbb{N}$ has the following properties :

(i) The chain has an unique stationary distribution $\pi = (\pi_1, \ldots, \pi_n)$.

(ii) For all $i, j \in S$ is $\lim_{t \to \infty} P_{j,i}^t < \infty$ and it is independent of j.

(iii) For all $i \in S$ we have $\pi_i = \lim_{t \to \infty} P_{j,i}^t = \frac{1}{h_{i,i}}$.

Proof: For all $i, j \in S$ the irreducibility of the chain implies that $\sum_{t \geq 1} r_{j,i}^t = 1$, and furthermore, for all $\varepsilon > 0$ exists a $t_1 = t_1(\epsilon) \in \mathbb{N}$ so that $\sum_{t=1}^{t_1} r_{j,i}^t \geq 1 - \varepsilon$ holds. For $i \neq j$, we have $P_{j,i}^t = \sum_{k=1}^{t} r_{j,i}^k P_{i,i}^{t-k}$. For $t \geq t_1$, we have $\sum_{k=1}^{t_1} r_{j,i}^k P_{i,i}^{t-k} \leq \sum_{k=1}^{t} r_{j,i}^k P_{i,i}^{t-k} = P_{j,i}^t$. By applying Lemma **??** and $t_1 \in \mathbb{N}$ we get:

$$\begin{aligned}
\lim_{t \to \infty} P_{j,i}^t &\geq \lim_{t \to \infty} \sum_{k=1}^{t_1} r_{j,i}^k P_{i,i}^{t-k} \\
&= \sum_{k=1}^{t_1} r_{j,i}^k \lim_{t \to \infty} P_{i,i}^t \\
&= \lim_{t \to \infty} P_{i,i}^t \sum_{k=1}^{t_1} r_{j,i}^k \\
&= (1 - \varepsilon) \lim_{t \to \infty} P_{i,i}^t
\end{aligned}$$

Similarly, we have $P_{j,i}^t = \sum_{k=1}^{t} r_{j,i}^k P_{i,i}^{t-k} \leq \sum_{k=1}^{t_1} r_{j,i}^k P_{i,i}^{t-k} + \varepsilon$ and it follows:

$$\begin{aligned}\lim_{t\to\infty} P_{j,i}^t &\leq \lim_{t\to\infty} \sum_{k=1}^{t_1} r_{j,i}^k P_{i,i}^{t-k} + \varepsilon \\ &= \sum_{k=1}^{t_1} r_{j,i}^k \lim_{t\to\infty} P_{i,i}^{t-k} + \varepsilon \\ &\leq \lim_{t\to\infty} P_{i,i}^t + \varepsilon\end{aligned}$$

With $\varepsilon \to 0$ and Lemma ?? we have

$$\lim_{t\to\infty} P_{j,i}^t = \lim_{t\to\infty} P_{i,i}^t = \frac{1}{h_{i,i}}$$

for all pairs $i, j \in S$.

Now let $\pi_i = \lim_{t\to\infty} P_{i,i}^t = \frac{1}{h_{i,i}}$. Next, we show that $\pi = (\pi_1, \ldots, \pi_n)$ is a stationary distribution. For any $t \geq 0$ and $i \in S$, we have $P_{i,i}^t \geq 0$ and thus $\pi_i \geq 0$. Further, for any $t \geq 0$ and any $j \in S$, we have $\sum_{i=1}^{n} P_{j,i}^t = 1$ and thus

$$\lim_{t\to\infty} \sum_{i=1}^{n} P_{j,i}^t = \sum_{i=1}^{n} \lim_{t\to\infty} P_{j,i}^t = \sum_{i=1}^{n} \pi_i = 1$$

which implies that π is a proper distribution. Through $P_{j,i}^{t+1} = \sum_{k=1}^{n} P_{j,k}^t P_{k,i}$ and with $t \to \infty$ we get

$$\pi_i = \sum_{k=1}^{n} \pi_k P_{k,i}$$

which shows that π is a stationary distribution.

To show the uniqueness, we suppose there is another stationary distribution ϕ. So we have $\phi_i = \sum_{k=1}^{n} \phi_k P_{k,i} = \sum_{k=1}^{n} \phi_k P_{k,i}^t$ which implies, with $t \to \infty$, that

$$\phi_i = \sum_{k=1}^{n} \phi_k \pi_i = \pi_i \sum_{k=1}^{n} \phi_k = \pi_i$$

for all $i \in S$. □

2.4.2 Random walk on a graph[12]

Let $G = (V, E)$ be a graph and $X_i : \Omega \to V$ a random variable for all $i \in \mathbb{N}$. A *random walk* on a graph with start distribution p is a sequence $X_1 X_2 X_3 \ldots$ of vertices that starts at a by p randomly chosen vertex X_1. The next vertex X_2 is chosen at random from the neighbours of X_1, analogue X_3 is a random neighbour of X_2, and so on. The transition probabilities are

$$P_{v,w} = \begin{cases} \frac{1}{deg(v)} & , \text{ if } w \in N(v) \\ 0 & , \text{otherwise} \end{cases}$$

[12] See: [**?**] and [**?**]

for all $v, w \in V$. If we stay at $v \in V$ and allow a move from v to v then the transition probability from v to $w \in V$ is

$$P_{v,w} = \begin{cases} \frac{1}{deg(v)+1} & \text{, if } w \in N[v] \\ 0 & \text{, otherwise} \end{cases}.$$

Remark:
A random walk on a graph is a little less general than finite[13] state, discrete time Markov chains that satisfy $p_v P_{v,w} = p_w P_{w,v}$[14] for all $v, w \in V$. Because, if we attach weights to the edges of the graph and allow loops then every finite state, discrete time Markov chain that satisfies $p_v P_{v,w} = p_w P_{w,v}$ can be built.

Lemma 2.12:
A random walk on an undirected graph G is aperiodic if and only if G is not bipartite.

Proof: In an undirected graph, there is always a walk of length two from a vertex to itself. If a graph B is bipartite then it does not have cycles with an ode number of edges, and so a random walk on B has the period $d = 2$. If the graph G is not bipartite then it has a cycle of odd length and by traversing this cycle we have a cycle of odd length from any vertex to itself. Ultimately a random walk on G is aperiodic. □

Theorem 2.13:
A random walk on a graph $G = (V, E)$ converges to a stationary distribution π, where

$$\pi_v = \frac{deg(v)}{2|E|}$$

for all $v \in V$.

Proof: Since $\sum_{v \in V} deg(v) = 2|E|$, it follows that

$$\sum_{v \in V} \pi_v = \sum_{v \in V} \frac{deg(v)}{2|E|} = 1$$

and so π is a proper distribution over V. Let P be the transition matrix of the Markov chain. The relation $\pi = \pi P$ implies

$$\pi_v = \sum_{w \in N(v)} \frac{deg(w)}{2|E|} \frac{1}{deg(w)} = \frac{deg(v)}{2|E|}$$

for all $v \in V$ and so the theorem follows. □

Corollary 2.14:
For any vertex $v \in V$ of a graph $G = (V, E)$ we have $h_{v,v} = \frac{2|E|}{deg(v)}$ by applying Theorem **??**.

[13] More general, it is enough to satisfy the locally finiteness which means that every state has a finite number of states it can move to, see [**?**].

[14] A Markov chain that satisfies $p_v P_{v,w} = p_w P_{w,v}$ is called reversible.

2.4.3 Random walk on a hypergraph

A random walk on a hypergraph $H = (V, E)$ with start distribution p is a sequence of vertices like a random walk on a graph. We introduce two models whereby the first is used in [**?**] and more in general in [**?**]. The first model, see [**?**] or [**?**], describes the walk as follows. Start the random walk at a, by p randomly chosen, vertex v. Then choose an edge that contains v with a probability proportional to $w(e)$[15] where $w(e)$ is a positive weight on the hyperedges. Next, choose a vertex $w \neq v$ uniformly at random from this edge, then repeat this process for w, and so on. The transition probability for any $v, w \in V$ is:

$$P_{v,w} = \frac{1}{\sum\limits_{e \in E} w(e)\,[v \in e]} \sum_{e \in E} w(e)\,[v \in e]\,\frac{[w \in e]}{r(e) - 1}$$

If we stay at $v \in V$ and allow a move from v to v then the transition probability is

$$P_{v,w} = \frac{1}{\sum\limits_{e \in E} w(e)\,[v \in e]} \sum_{e \in E} w(e)\,[v \in e]\,\frac{[w \in e]}{r(e)}$$

for all $w \in V$.

Theorem 2.15:
A random walk on a hypergraph $H = (V, E)$ converges to a stationary distribution π, where

$$\pi_v = \frac{\sum\limits_{e \in E} w(e)\,[v \in e]}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)\,[w \in e]}$$

for all $v \in V$.

Proof: Let $H = (V, E)$ be a hypergraph. We have:

$$\begin{aligned}
\sum_{v \in V} \pi_v &= \sum_{v \in V} \frac{\sum\limits_{e \in E} w(e)[v \in e]}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)[w \in e]} \\
&= \frac{1}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)[w \in e]} \sum_{v \in V} \sum_{e \in E} w(e)\,[v \in e] \\
&= 1
\end{aligned}$$

It follows:

$$\begin{aligned}
\sum_{v \in V} \pi_v P_{v,w} &= \sum_{v \in V} \frac{\sum\limits_{e \in E} w(e)[v \in e]}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)[w \in e]} \cdot \frac{1}{\sum\limits_{e \in E} w(e)[v \in e]} \sum_{e \in E} w(e)\,[v \in e]\,\frac{[w \in e]}{r(e)} \\
&= \frac{1}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)[w \in e]} \sum_{v \in V} \sum_{e \in E} w(e)\,[v \in e]\,\frac{[w \in e]}{r(e)} \\
&= \frac{1}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)[w \in e]} \sum_{e \in E} w(e)\,[w \in e] \sum_{v \in V} \frac{[v \in e]}{r(e)} \\
&= \frac{1}{\sum\limits_{w \in V} \sum\limits_{e \in E} w(e)[w \in e]} \sum_{e \in E} w(e)\,[w \in e] \\
&= \pi_w
\end{aligned}$$

So the theorem is shown. □

[15] In [**?**] is $w(e) = r(e) - 1$ or $w(e) = 1$ for all $e \in E$

The second model is a random walk on $\mathrm{bipart}(H)$. It is a random walk on a graph which is discussed at Section ??.

2.4.4 Simulated Annealing and Random walk[16]

This section describes the relation of the simulated annealing algorithm and its induced random walk. The simulated annealing algorithm deals with the topic that we have an objective function f that should be minimized. We denote the admissible solution, for which f is minimal, with x^*. The algorithm is based on a principle of metallurgy to achieve a clean crystal structure. That is, at first, heating the substance in addition to its melting point, then cooling the substance gently to get a good crystal structure which means to get a global energy minimum for f. If we cool to fast we just get a local energy minimum for f. Observe that we minimize locally if we go from an admissible solution x to a better admissible solution y if it is possible. If there are not better solutions then we achieve a minimum. But it could be a local minimum. So, to have a chance to leave a local minimum, we accept with a certain probability a worse solution. This probability decreases over time and simulates the temperature of the simulated annealing process.

Remark:
If the temperature $T > 0$ is fix, the simulated annealing process is transferred to the metropolis algorithm. If $T = 0$ all over the time, the process is transferred to the local random search algorithm. Furthermore, we do not distinguish between those algorithms and call all of them simulated annealing, instead.

To handle this algorithmically, we assume that for every x in the solution space exists an environment $U(x)$ for all $T \geq 0$ so that the following condition holds:

(i) For all x we have $x \in U(x)$.

(ii) For all x, y we have $x \in U(y) \Leftrightarrow y \in U(x)$.

The simulated annealing algorithm is:

[16] See: [**?**]

Algorithmus 3: SA

Input: A temperature $T_0 \geq 0$ and a start solution x_0.

Output: Suboptimal solution x.

```
x = x_0, T = T_0
repeat
    Choose uniformly at random y ∈ U(x)
    if f(y) ≤ f(x) then
        x = y
    else
        Set x = y with probability e^{-(f(y)-f(x))/T}
    end if
    Update T
until The repeats are done sufficiently often
return x
```

Let $T \geq 0$ and x an admissible solution. So, $G_T(x,y)$ is the probability to choose $y \in U(x)$ and $A_T(x,y)$ is the probability that we accept y. We have:

$$G_T(x,y) = \begin{cases} \frac{1}{|U(x)|} & \text{, if } y \in U(x) \\ 0 & \text{, otherwise} \end{cases} \text{ and } A_T(x,y) = \begin{cases} 1 & \text{, if } f(y) \leq f(x) \\ e^{-\frac{f(y)-f(x)}{T}} & \text{, otherwise} \end{cases}$$

Then the probability that we change from x to y is

$$P_T(x,y) = \begin{cases} G_T(x,y)A_T(x,y) & \text{, if } x \neq y \\ 1 - \sum\limits_{\substack{z \in U(x) \\ z \neq x}} G_T(x,z)A_T(x,z) & \text{, if } x = y \end{cases}.$$

Corollary 2.16:

The simulated annealing algorithm performs, on a graph (V,E) whose vertices are the admissible solutions and whose edge set is $E = \{\{x,y\} \mid x \in V, y \in U(x)\}$, a random walk with transition probabilities $P_T(x,y)$ for all x,y and a fixed temperature $T \geq 0$, otherwise the random walk is not homogeneous. On the other hand, a random walk with transition probability

$$P_{v,w} = \begin{cases} \frac{1}{deg(v)+1} & \text{, if } w \in N[v] \\ 0 & \text{, otherwise} \end{cases}$$

is a realization of a simulated annealing algorithm on the vertex set of the random walk with $U(v) = N[v]$ for all $v \in V$ and a fixed temperature $T \geq 0$.

3 Hierarchical structure[1]

This section provides the principle, which will be applied in the next section, in an abstract way. A *category*[2] $\mathbf{C}$ is a pair $(\mathcal{C}_0, \mathcal{C}_1)$ that consists of a class of objects $\mathcal{C}_0$ and a class of morphisms between the objects $\mathcal{C}_1$, called *arrows*, where a category has to satisfy the following axioms:

(i) For each $f \in \mathcal{C}_1$ there are two given objects $\mathrm{dom}(f) \in \mathcal{C}_0$ and $\mathrm{codom}(f) \in \mathcal{C}_0$ called the *domain* and *codomain* of f.

(ii) For any two arrows $f : A \to B$ and $g : B \to C$ with $\mathrm{codom}(f) = \mathrm{dom}(g)$ exists an arrow

$$g \circ f : A \to C$$

which is called the *composition* of f and g.

(iii) For each object $A \in \mathcal{C}_0$ there is an arrow

$$1_A : A \to A$$

which is called the *identity arrow* of A.

(iv) The composition is *associative* which means that

$$h \circ (g \circ f) = (h \circ g) \circ f$$

holds for any three arrows $f, g, h \in \mathcal{C}_1$.

(v) For all arrows $f : A \to B$ there is a *right unit* 1_A and a *left unit* 1_B so that

$$f \circ 1_A = f = 1_B \circ f$$

holds.

Some statements about categories could be visualized by a diagram.

Example 3.1:

The left side of Figure ?? shows a diagram of two arrows $f : A \to B$, $g : B \to C$, and their composition $g \circ f$.
The right side of this figure shows the diagram of the associativity of three arrows $f : A \to B$, $g : B \to C$, and $h : C \to D$.

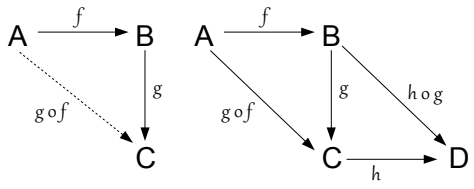


Figure 3.1: Diagram of the composition of two arrows and the associativity of three arrows.

[1] See: [**?**]
[2] See: [**?**]

Remark:
A category could also be imaged as a generalization of a monoid[3] because the arrows of every category with exactly one object build a monoid with respect to the composition.

We call a category $\mathbf{C} = (\mathcal{C}_0, \mathcal{C}_1)$ *discrete or finite object category* if $\mathcal{C}_0$ is countably or finite. Also, we call $\mathbf{C}$ *discrete or finite category* if $\mathcal{C}_1$ is countably or finite. Further, we call $\mathbf{C}$ *locally finite* if there is just a finite number of arrows between any two objects in $\mathbf{C}$.

Example 3.2:
Let us now consider some examples:

(i) We consider some categories with sets as objects. Category $\mathbf{Sets}$ is the category where the objects are sets and the arrows are the functions between this sets. Category $\mathbf{Sets}_{fin}$ is the category of all finite sets and functions between them. Further, category $\mathbf{Sets}^*_{fin}$ is the category of all finite sets and all injective functions between them. And so on.

(ii) We consider sets equipped with a binary relation $\leq$. If A is a set equipped with the binary relation $\leq_A$ and satisfies

a) reflexivity: $a \leq_A a$

b) transitivity: $a \leq_A b$ and $b \leq_A c$ then $a \leq_A c$

c) antisymmetry: If $a \leq_A b$ and $b \leq_A a$ then $a = b$.

for all $a, b, c \in A$ then A is called partial ordered set, or posets for short. Further, we call relations that satisfy a), b), and c) an *order*. The category $\mathbf{POS}$ is the category of all posets and arrows between them. An arrow from a poset A to a poset B is a monotonic function $m : A \to B$ which means that $a \leq_A a' \Rightarrow m(a) \leq_B m(a')$ holds for all $a, a' \in A$.

(iii) We consider some finite categories like the category $\mathbf{0}$. This category has no objects and no arrows. Category $\mathbf{1}$ has one object and its identity arrow. Category $\mathbf{2}$ has two objects, their identity arrows, and one arrow from one object to the other one. Category $\mathbf{3}$ has three objects, their identity arrows, and the other arrows build a commute triangle. Figure ?? shows the diagram of this three categories where the identity arrows are not drawn.

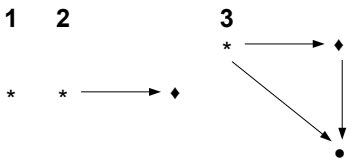


Figure 3.2: Diagram of the categories $\mathbf{1}, \mathbf{2}$, and $\mathbf{3}$.

(iv) We consider the categories of proofs in which the objects are formulas and an arrow from a formula φ to a formula ψ is a deduction of ψ from the assumption φ. Note that there can

[3] A semigroup with unit

be many different arrows $p : \varphi \to \psi$ if there are different deductions from φ to ψ.

(v) We consider the category of categories. The objects are categories and the arrows are so called functors. A *functor* $F : \mathbf{C} \to \mathbf{D}$ between two categories $\mathbf{C} = (\mathcal{C}_0, \mathcal{C}_1), \mathbf{D} = (\mathcal{D}_0, \mathcal{D}_1)$ is a mapping that maps $\mathcal{C}_0$ to $\mathcal{D}_0$ and $\mathcal{C}_1$ to $\mathcal{D}_1$ so that

- $F(f : A \to B) = F(f) : F(A) \to F(B)$
- $F(1_A) = 1_{F(A)}$
- $F(g \circ f) = F(g) \circ F(f)$

holds for all $f, g \in \mathcal{C}_1$ and all identity arrows $1_A \in \mathcal{C}_1$.

Let $\mathbf{C} = (\mathcal{C}_0, \mathcal{C}_1)$ be a category, then the *dual* or *opposite* category $\mathbf{C}^{op}$ of $\mathbf{C}$ is the category with the same objects as $\mathbf{C}$, but an arrow $f : B \to A$ in $\mathbf{C}^{op}$ is an arrow $f : A \to B$ in $\mathbf{C}$. Formally it means that $\mathbf{C}^{op} = (\mathcal{C}_0, \{f' : B \to A \mid f : A \to B \in \mathcal{C}_1\})$. It is easy to see that we get $\mathbf{C}^{op}$ out of the category $\mathbf{C}$ if we replace

(i) codom for dom,

(ii) dom for codom and

(iii) $f \circ g$ for $g \circ f$

in $\mathbf{C}$. A category $\mathbf{C} = (\mathcal{C}_0, \mathcal{C}_1)$ is called *small* if $\mathcal{C}_0$ and $\mathcal{C}_1$ are sets. Otherwise, $\mathbf{C}$ is called *large*. Let I be an index set and $f_i \in \mathcal{C}_1$ for all $i \in I$, so we call $(f_i)_{i \in I}$ a *series of arrows in* $\mathbf{C}$ if $\mathrm{codom}(f_i) = \mathrm{dom}(f_{s(i)})$ holds for all $i \in I$ where $s(i)$ is the successor of $i \in I$.

Corollary 3.3:
Let A be a series of arrows in $\mathbf{C}$. Then the category $\mathbf{A}$, whose objects are the objects of A and whose arrows are the arrows of A, the composition of this arrows, and the identity arrows of the objects of A, is small.

The first element of a series of arrows is called *initial arrow* of this series. The series of arrows is called *finite* if the index set $|I| \in \mathbb{N}$ and the arrows of the series satisfy $\mathrm{codom}(f_i) = \mathrm{dom}(f_{s(i)})$ for all $i \in I$ that has a successor. Further, the last element of a finite series of arrows is called *terminal arrow* of this finite series.

Remark:
A series of arrows $(f_i)_{i \in I}$ in $\mathbf{C}$ implies a series of objects that starts with the domain of the initial arrow of this series, then follow the codomain of the initial arrow of this series, then the codomain of the successor of the initial arrow, and so on. Further, different series of arrows with the same initial arrow could lead to the same series of objects.

A series of arrows with index set I is called *random* if each successor of an element f_i with $i \in I$ of this series is choosen randomly from all arrows that have $\mathrm{codom}(f_i)$ as domain.

Lemma 3.4:
Let $\mathbf{C} = (\mathcal{C}_0, \mathcal{C}_1)$ be a non empty, at most discrete category. Then there is a random walk $\mathcal{R}(\mathbf{C})$ with time I and state set $\mathcal{C}_0$ on the directed graph that is inducted by the category $\mathbf{C}$.

Proof: Let $\Omega = \{(A, f) \mid A \in \mathcal{C}_0, f : A \to B \in \mathcal{C}_1\}$ be the sample space of the moves between to objects in $\mathbf{C}$. Further, the category $\mathbf{C}$ inducts a directed graph

$$G = (\mathcal{C}_0, \{(A, B) \mid f \in \mathcal{C}_1, f : A \to B\})$$

where (A, B) denotes an arc from A to B. While $\mathcal{C}_1$ is at most countably there exists by the axioms of Kolmogorov a probability distribution $p(\omega)$ over Ω so that

$$\begin{aligned}\sum_{\omega\in\Omega} Pr(\omega) &= \sum_{(A,f)\in\Omega} Pr(\text{be in } A,\ \text{move from } A \text{ to } B \text{ among } f)\\ &= \sum_{(A,f)\in\Omega} Pr(\text{be in } A)\, Pr(\text{move from } A \text{ to } B \text{ among } f)\\ &= \sum_{A\in\mathcal{C}_0} Pr(\text{be in } A) \sum_{f\in\mathcal{C}_1} Pr(\text{move from } A \text{ to } B \text{ among } f)\\ &= 1.\end{aligned}$$

We want to consider only distributions which satisfy

$$\sum_{f\in\mathcal{C}_1} Pr(\text{move from } A \text{ to } B \text{ among } f) = 1 \tag{3.1}$$

for all $A \in \mathcal{C}_0$. This distribution exists because the category is not empty and so there is $A \in \mathcal{C}_0$, and for all $A \in \mathcal{C}_0$ the identity arrow $1_A : A \to A$ is in $\mathcal{C}_1$ so we have for example the distribution

$$Pr(f) = \begin{cases} 1 & \text{, if } f = 1_A \\ 0 & \text{, otherwise} \end{cases}$$

for all $f \in \mathcal{C}_1$ which satisfies $(??)$. This implies the random walk $\mathcal{R}(\mathbf{C}) = (X_i)_{i\in I}$ with

$$X_i : \Omega \to S : (A, f : A \to B) \mapsto \text{codom}(f)$$

for all distributions which satisfy $(??)$. So, X_i is the i-th random motion which starts at an object A and moves among f to an object B. It is obviously clear that the Markov probability

$$Pr(X_t = B \mid X_{t-1} = A, X_{t-2} = A_{t-2}, \ldots, X_0 = A_0) = Pr(X_t = B \mid X_{t-1} = A)$$

with $t \in I$ and $A, B, A_{t-2}, \ldots, A_0 \in S$ holds. Further, we have the transition probabilities

$$P_{A,B} := \sum_{f:A\to B\in\mathcal{C}_1} Pr(\text{move from } A \text{ to } B \text{ among } f)$$

for all $A, B \in \mathcal{C}_0$. □

Corollary 3.5:
Every random series in a nonempty and at most discrete category $\mathbf{C}$ is a realization or a part of a realization of a random walk. Further, if we consider an objective function over the objects in $\mathbf{C}$ then a random series of arrows is a realization of the simulated annealing algorithm, see Section $??$, whose neighbourhood selections are given by the series.

We call a series of arrows $(f_i)_{i\in I}$, with an index set I, a *hierarchical structure* with respect to an ordering $\leq$ if $\mathrm{dom}(f_i) \leq \mathrm{codom}(f_i)$, f_i is called an *up arrow*, or $\mathrm{codom}(f_j) \leq \mathrm{dom}(f_j)$, f_j is called a *down arrow*, holds for all $i, j \in I$. Further, an arrow f_i with $i \in I$ is called *stay arrow* if $\mathrm{codom}(f_i) = \mathrm{dom}(f_i)$ holds. We call a random series of arrows *random hierarchical structure* respective to an ordering $\leq$ if this random series of arrows satisfies the restrictions with respect to the odering $\leq$. A hierarchical structure $(f_i)_{i\in I}$ with respect to $\leq$ is called *bounded above* by an object A if there is not an index $i \in I$ with $A \leq \mathrm{codom}(f_i)$ and $A \neq \mathrm{codom}(f_i)$. Further, a hierarchical structure $(f_i)_{i\in I}$ with respect to $\leq$ is called *bounded below* by an object A if there is not an index $i \in I$ with $\mathrm{codom}(f_i) \leq A$ and $A \neq \mathrm{codom}(f_i)$. We continue with two examples of hierarchical structures.

Example 3.6:

At first, we consider the hierarchical structure of a multigrid method[4] that will be applied in numerical analysis to solve linear equations which occur by discretisation[5] of partial differential equations, for example. Let O be the differential problem or boundary value problem we want to solve. For example:

$$O \begin{cases} \Delta u(x) = f(x) \\ x \in \Omega \\ u|_{\delta\Omega} = g \end{cases}$$

The discretisation of O, which we consider, means that just some of the points in Ω will be regarded. They are chosen by a step width $h \in \mathbb{R}$ and the set of all of them will be denoted with Ω_h. The set Ω_h is called a *grid*. For example, if $\Omega = \{x \mid a \leq x \leq b\}$ then Ω_h could be $\{x_i \mid x_i = a + ih, i = 0, 1, \ldots, n\}$ with $h = \frac{b-a}{n}$ and $n \in \mathbb{N}$. Further, the discretisation of O delivers the following problem

$$O_h \begin{cases} A_h u_h(x) = b_h \\ x \in \Omega_h \\ u_h(x) = g, \text{ if } x \in \delta\Omega \end{cases}$$

with $A_h \in \mathbb{R}^{n\times n}$, $b_h \in \mathbb{R}^n$, $n \in \mathbb{N}$, step width $h \in \mathbb{R}$ and $u_h \in \mathbb{R}^n$. We want to achieve that $u_h^T \approx (u(x_1), \ldots, u(x_n))$ with $x_1, \ldots, x_n \in \Omega_h$. The solution of the linear equation in O_h is denoted by u^*. For every $u_h \in \mathbb{R}^n$ we have an error $e_h = u^* - u_h$ that implies the residual equation $A_h e_h = A_h(u^* - u_h) = b_h - A_h u_h = r_h$ with residuum $r_h \in \mathbb{R}^n$. If we know the residuum r_h, we know the error through $e_h = A_h^{-1} r_h$ and so the solution $u^* = u_h - e_h$. To calculate this error with respect to the residuum is as hard as to calculate the solution. That means, we get in trouble if the dimension of A_h is large. This is where the multigrid method comes in use, for example. So we transform the current grid Ω_h that implies A_h, r_h to a coarser grid $\Omega_{h'}$ that implies $A_{h'}, r_{h'}$. We denote this transformation by $r : (A_h, r_h) \to (A_{h'}, r_{h'})$. Further, we have $A_{h'} = r_1(A_h, r_h)$ where $A_{h'}$ and A_h only differ in their order, and $r_{h'} = r_2(A_h, r_h)$ where $r_h, r_{h'}$ only differ in their

[4] See: [**?**] page 354 et seqq.

[5] See: [**?**] page 32

order, too. The transformation r describes some restrictions on Ω_h and so this transformation is called *restriction*. We get a new problem with this restriction:

$$A_{h'}e_{h'} = r_{h'}$$

So, this linear system is easier to solve because it has a lower order. If we have solution $e_{h'}$ we transform it back to the finer grid Ω_h with step width h. This transformation is called *prolongation* and could be done by an interpolation. Let $p : (A_{h'}, e_{h'}) \to (A_h, e_h)$ describe the prolongation. Further, we have $A_h = p_1(A_{h'})$ and $p_2(e_{h'}) = e_h$. So we have an approximation $e_h^{new} = p(e_{h'})$ of the error which is implied by the problem O_h. This implies an approximation $u_h^{new} = u_h + e_h^{new}$ of u^*. This process is called a two grid process. The whole process could also be restated with $u_h = u_h^{new}$. Further, we could use more than two grids. For example, a third grid could be used to approximate the error which is used to approximate the solution. More in general, we build the category $\mathbf{MatVec}(n,m)$, $\mathbf{MV}(n,m) = (\mathcal{MV}_0^{n,m}, \mathcal{MV}_1^{n,m})$ for short, with $n, m \in \mathbb{N}$. It consists of all pairs (A, r) of matrices $A \in \mathbb{R}^{k\times l}$ and vectors $v \in \mathbb{R}^k$ with $k \leq n, l \leq m$ and a class of arrows which we want to distinguish in three types. The first type of this arrows is an arrow $r : (A, v) \to (B, w)$ from (A, v) to (B, w) that occurs if B is a submatrix of A and w is a subvector of v. The second type is an arrow $p : (B, w) \to (A, v)$ from (B, w) to (A, v) that occurs if B is a submatrix of A and w is a subvector of v. The last type is an arrow $s : (A, v) \to (A, w)$ from (A, v) to (A, w) that occurs if v and w have the same order. Note that the conditions on a category hold for $\mathbf{MatVec}(n,m)$. That means, that $\mathbf{MatVec}(n,m)$ is closed under composition, that is, for any $a, b \in \mathcal{MV}_1^{n,m}$ we have $a \circ b \in \mathcal{MV}_1^{n,m}$. Further, for every object (A, v) in $\mathbf{MatVec}(n,m)$ there is an identity arrow $s : (A, v) \to (A, v)$. Finally, the arrows are associative, that is, for any three arrows $a, b, c \in \mathcal{MV}_1^{n,m}$ we have $(a \circ b) \circ c = a \circ (b \circ c)$. It follows that any multigird method is a series of arrows in $\mathbf{MatVec}(n,n)$ where n is the order of A_h of O_h for any step width h. For example, for the two grid method we have

$$(A_h, r_h) \xrightarrow{r} (A_{h'}, r_{h'}) \xrightarrow{s} (A_{h'}, e_{h'}) \xrightarrow{p} (A_h, e^{new})$$

which implies the series of arrows r, s, p. The two grid method is a scheme which is called V-cycle. Two typical cycle schemes are shown in Figure ??.

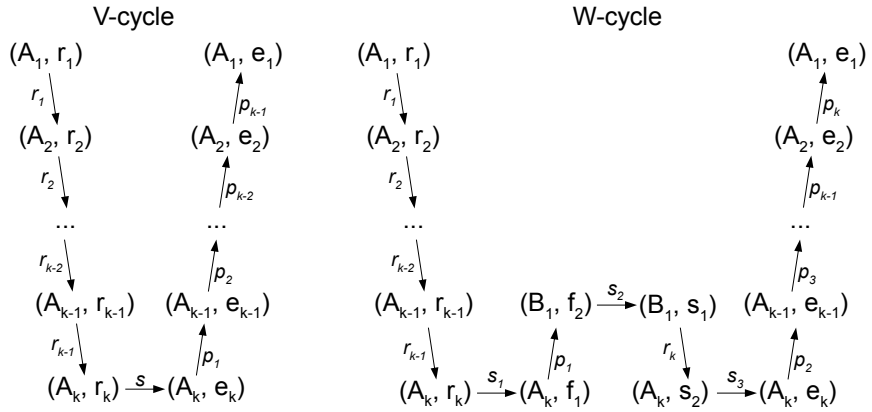


Figure 3.3: V-cycle scheme is shown on the left and the so called W-cycle scheme is shown on the right.

The down arrows of this arrow series are arrows of type r, the stay arrows are of type s, and the up arrows are of type p. Further, the considered arrow series is a hierarchical structure respective to the ordering $\leq$ where $(A, v) \leq (B, w)$ if A is a submatrix of B and v is a subvector of w for $(A, v), (B, w) \in \mathcal{MV}_1^{n,m}$. By Lemma **??** follow that if all arrows point from (A, v) to (B, w) in a random fashion then the hierarchical structure is a random hierarchical structure. Further, if we consider an objective function on the objects which should be minimized by the random hierarchical structure, this structure is a realization of a proper simulated annealing algorithm.

Example 3.7:

In this example, we consider the hierarchical structure of the hierarchical partitioning of hypergraphs. Let $\mathbf{HypPart}$ be the category of all pairs of finite hypergraphs H and all subsets of the power set of $V(H)$. The morphisms $+, \hat{+}, -, \hat{-}, /, \cdot_W$ of Section **??**, where $\cdot$ is a place holder of an arbitrary hypergraph, and $p : (H, M) \to (H, M')$ with $M, M' \subseteq 2^{V(H)}$ are the arrows of $\mathbf{HypPart}$. The meaning of this arrows is the following:

$(\mathbf{H}, \mathbf{M}) \xrightarrow{+} (\mathbf{H'}, \mathbf{M})$**:** There is a set $W \subseteq V(H')$ so that $H + W \simeq H'$.

$(\mathbf{H}, \mathbf{M}) \xrightarrow{\hat{+}} (\mathbf{H'}, \mathbf{M})$**:** There is a set $F \subseteq E(H')$ so that $H \hat{+} F \simeq H'$.

$(\mathbf{H}, \mathbf{M}) \xrightarrow{-} (\mathbf{H'}, \mathbf{M})$**:** There is a set $W \subseteq V(H)$ so that $H - W \simeq H'$.

$(\mathbf{H}, \mathbf{M}) \xrightarrow{\hat{-}} (\mathbf{H'}, \mathbf{M})$**:** There is a set $F \subseteq E(H)$ so that $H \hat{-} F \simeq H'$.

$(\mathbf{H}, \mathbf{M}) \xrightarrow{\mathbf{W}} (\mathbf{H'}, \mathbf{M})$**:** There is a set $W \subseteq V(H)$ so that $H_W \simeq H'$.

$(\mathbf{H}, \mathbf{M}) \xrightarrow{\backslash} (\mathbf{H'}, \mathbf{M})$**:** There is a set $F \subseteq E(H)$ so that $H \backslash F \simeq H'$.

The *hierarchical partitioning of a hypergraph* is a hierarchical structure respective to the ordering $\leq$ where $(H', M) \leq (H, M)$ if H' is a subhypergraph or minor of H. We can also perform cycle schemes as we see in the example before. The most common cycle scheme is the V-cycle scheme, because for a W-cycle scheme we need an iterative behaviour like the error

approximation. But, if we go down among some arrows, partitioning the achieved hypergraph, go up among few arrows, again go down among some arrows to (H', M), and partition H' to (H', M') is the same as to go down to (H', M) and partition them. The W-cycle could be applied if we use the information of the partition M when we go down again. We do not consider such strategies in this thesis. A typical V-cycle is shown in Figure ??.

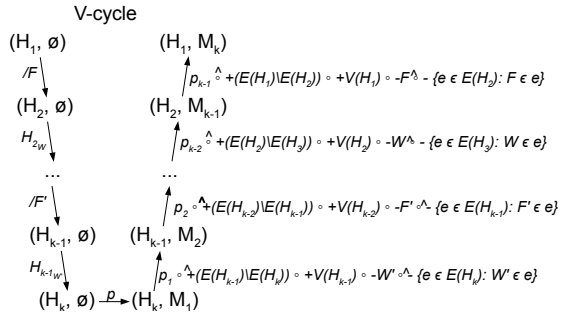


Figure 3.4: V-cycle scheme of a hierarchical partitioning.

If the arrows of the hierarchical structure act in random fashion then the hierarchical partition is a random hierarchical partition. Further, if we consider an objective function on the objects of the random hierarchical partition then this partition is a realization of a proper simulated annealing algorithm, by Corollary ??.

It is easy to see that all hierarchical structures imply a partial ordered set. Let $\mathbf{Hir}$ be the category of all hierarchical structures where the arrows are the monotonic functions of the inducted posets in category $\mathbf{POS}$. Further, let $\mathbf{SA}$ be the category of all random hierarchical structures equipped with an objective function per random hierarchical structure. The arrows are the monotonic functions as in $\mathbf{Hir}$. So, $\mathbf{SA}$ is the category of all simulated annealing realizations respective to the objective functions of the random structures.

4 Hierarchical partitioning

In this section, we deal with the hierarchical partitioning problem of an IC-hypergraph. Example ?? introduces this topic, but we just need a few arrows of the category $\mathbf{HypPart}$. Further, we adapt the names of the three arrow types of this category as follows:

down arrow	$\leftrightarrow$	coarsening
stay arrow	$\leftrightarrow$	partitioning or improvement
up arrow	$\leftrightarrow$	refinement

The next section introduces the hierarchical partitioning problem of an IC-hypergraph more in detail. The algorithms run at PC1[1] except the benchmark where the algorithms run at PC2.

4.1 Problem and modelling

The hierarchical partitioning problem of an IC-hypergraph is captured by the following question: *How should the IC-hypergraph be partitioned to get a good opportunity for a good placement and wiring of the IC?*
This question leads to a difficulty that occurs because we do not have the information of the placement and wiring at that step where we partition an IC-hypergraph. So, we can not decide which partitioning is better for the arrangement of the elements of the IC. Another difficulty is that the graph partitioning and thus the hypergraph partitioning is an $\mathcal{NP}$-hard problem[2]. For that, we use simple heuristics to get a suboptimal solution under some restrictions as we introduced in section ??. This solution respect the following two main goals for the placement and wiring:

(i) The area consumption should be as low as possible.

(ii) A signal should reach its destination at a given time.

Furthermore, we assume that the top-down strategy is used for the placement and wiring of the elements of the IC because the individual placement and wiring of the elements of a large IC is too time-consuming. The top-down strategy consists of two steps for placing and wiring the elements of the IC. At first, the elements will be partitioned into blocks and this blocks will be placed and wired on a board. Secondly, the elements in each block will be placed and wired on this part of the board where this block has been placed in the first step. At that, the IC should be partitioned in such a way that neither a block with nearly all elements nor a block that consumes the whole area of the board arises. Otherwise, it is too time-consuming to place

[1] See appendix.
[2] See: [**?**]

and wire the elements of this block. A partition $P = \{B_1, \ldots, B_k\}$ with $k \in \mathbb{N}$ blocks of a hypergraph is called *limited above by* $C \in \mathbb{R}$ if $\mu(B_i) \leq C$ holds for all $i \in [k]$ and it is called *limited below by* $c \in \mathbb{R}$ if $\mu(B_i) \geq c$ holds for all $i \in [k]$ where μ is a set function into $\mathbb{N}$. For instance, μ could be the number of elements or the area consumption of a block and so on. For further, we define μ as the area consumption of a block. A partition $P = \{B_1, \ldots, B_k\}$ with $k \in \mathbb{N}$ blocks of a hypergraph is called *balanced by* $C' \in \mathbb{R}$ if there is a $\beta \in [0, 0.5]$ so that $\beta C' \leq \mu(B_i) \leq (1 - \beta)C'$ holds for all $i \in [k]$. We optimize the partitioning with respect to an objective function that could be useful for the placement and wiring to achieve the goals (i) and (ii). One of the most commonly used objective functions is the minimum cut that is defined by

$$\mathcal{M}(H, P) = \sum_{e \in E(H)} [\text{there is a } B \in P \text{ so that } 0 < |e \cap B| < |e|]$$

where H is a partitioned hypergraph with partition $P = \{B_1, \ldots, B_k\}$. The minimization of the objective function $\mathcal{M}$ acts on H so that as few edges as possible appear between the blocks in P. Furthermore, few edges between the blocks lead to a small area consumption of the edges between the blocks and also it may lead to few interlayer connections in 3D-IC designs. But this approach has a big disadvantage because the structure of the block connectivity graph is ignored. The two examples, Example ?? and Example ??, show two of this disadvantages.

Example 4.1:
We have

$$\mathcal{M}(G, \{\{u\}, \{v\}, \{w\}, \{x\}\}) = \mathcal{M}(G', \{\{u\}, \{v\}, \{w\}, \{x\}\})$$

for the two graphs G, G' that are shown in Figure ??. So G is as good as G' with respect to the minimum cut. But, if we move u or x in G then we have to respect some restrictions, like the clock constraint, to the two neighbours v, w and if we move v or w then we have to respect restrictions to their three neighbours. On the other hand, if we move u, v, or x in G' then we only have to respect some restrictions to w and if we move w then we have to respect this restrictions to all three neighbours. As we see, there is a difference for the arrangement of the vertices from G and G' that is not respected by the minimization of the cut.

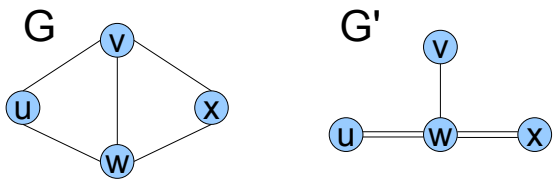


Figure 4.1: Two graphs G and G' with the same minimum cut but with different behaviour in motion.

Example 4.2:

Let H be a hypergraph that is partitioned two times into six blocks. The first partitioning leads to partition P and the second leads to partition Q. Let G be the edges weighted block connectivity graph of the partitioning P and let G' be the edge weighted block connectivity graph of the partitioning Q. Furthermore, the edge weight of both graphs of an edge $e = \{x, y\}$ is defined as the number of edges between the blocks $x \in [6]$ and $y \in [6]$ in H. Both graphs are shown in Figure **??**. We have:

$$\mathcal{M}(H, P) = 30 < 50 = \mathcal{M}(H, Q)$$

So, the partitioning P is better than the partitioning Q with respect to the minimum cut. But, if we perform a 3D placement with two layers in such a way that we place block $1, 2$, and 3 to layer one and block $4, 5$, and 6 to layer two then we have 18 interlayer connections with respect to partitioning P and 10 interlayer connections with respect to partitioning Q.

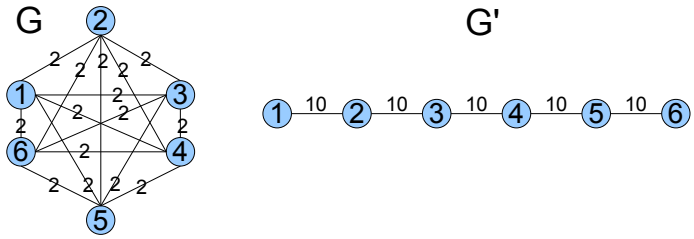


Figure 4.2: Two graphs G and G' demonstrate that the block connectivity graph has to be respected.

Furthermore, if we achieve few edges between the blocks then more edges are inside the blocks. In fact, more edge crossings are inside the blocks, but this is not a big problem if the board has enough metal layer to handle this. Also, this could lead to a higher area consumption of the blocks because the elements of an IC became smaller in the past, and thus the edges of a block consumes a larger amount of its total area. It follows that the minimum cut is not a favourable objective function to achieve the main goals. So, other approaches are used like the min-max approach of Ding et al[3]. Example **??** shows that it is unfavourable for the movement abilities of a block if it is connected with too many other blocks. Let H be a partitioned hypergraph with partition P. We call the edges of its block connectivity graph *dependencies*. So, two blocks depend on themselves if their corresponded vertices in the block connectivity graph are connected. Further, we could minimize the dependencies to achieve good movement abilities for the blocks. Formally it means:

$$E(\mathrm{BC}(H, P)) \to Min$$

Furthermore, Example **??** illustrates that few edges between the blocks could be useful, especially if the block connectivity graph has few dependencies. So, we define the following objective

[3]See: [**?**]

function

$$\mathcal{F}(H,P) = \frac{|E(\mathrm{BC}(H,P))|}{|V(\mathrm{BC}(H,P))| \cdot (|V(\mathrm{BC}(H,P))|-1)/2} + \alpha \cdot \frac{\mathcal{M}(H,P)}{|E(H)|} \to Min \qquad (4.1)$$

with $\alpha \in \mathbb{R}^+$ which should be minimized. We call $\frac{|E(\mathrm{BC}(H,P))|}{|V(\mathrm{BC}(H,P))|\cdot(|V(\mathrm{BC}(H,P))|-1)/2}$ the *dependency ratio* of H respective to P and $\frac{\mathcal{M}(H,P)}{|E(H)|}$ the *cut ratio* of H respective to P. The algorithms that minimize $\mathcal{F}(H,P)$ are tested on several of the IC-hypergraphs of the netlists of benchmark "ISPD 05/06"[4] in Section ??. But at first, we introduce the algorithm and apply them to the IC-hypergraph of the smallest netlist, which is called adaptec1, of this benchmark. Some data of the IC-hypergraph from the netlist adaptec1 is presented below.

Name	Number of vertices	Number of edges	δ	Δ	Average degree	s	r	Average number of vertices in an edge
adaptec1	211447	221142	1	288	≈ 4.35	1	1270	≈ 4.163

Figure ?? present the degree profile of the IC-hypergraph from the netlist adaptec1. This profile shows how many vertices of a certain degree occur. For a better clarity, we used a logarithmic scale for the y-axis.

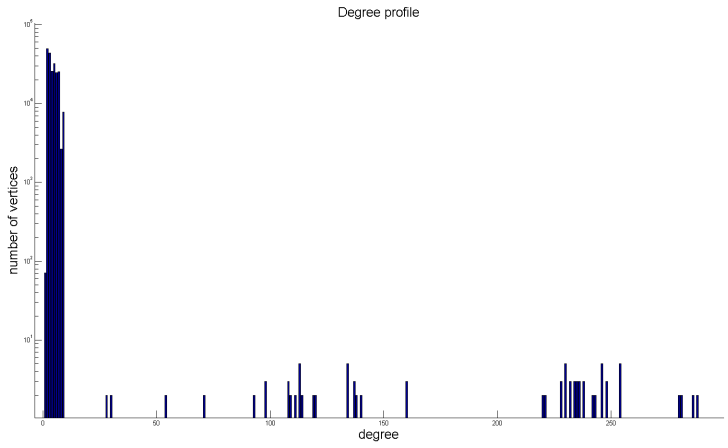


Figure 4.3: Degree profile from adaptec1.

In Figure ?? we see that the most vertices have degree of two or three. It is typical for IC-hypergraphs because the most elements of an IC are logical elements, like AND or OR and so on, and diodes. Another important profile is the edge profile. It shows how many edges of a certain size occur. This profile is presented in Figure ??. For a better clarity, we used a logarithmic scale for the x-axis and the y-axis.

[4] See: [**?**]

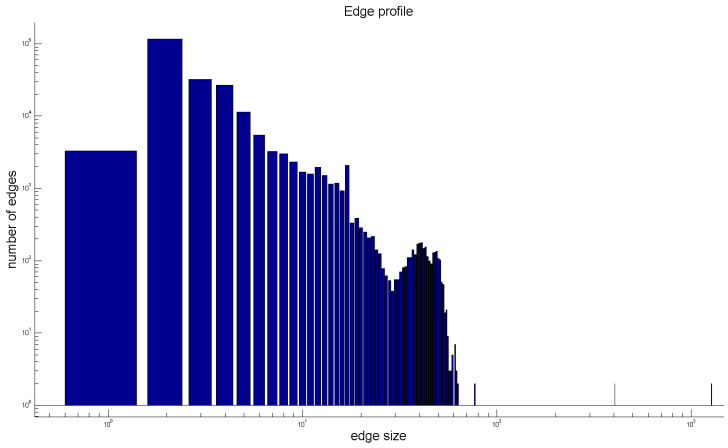


Figure 4.4: Edge profile from adaptec1.

The edge profile shown in Figure ?? is also typical for IC-hypergraphs where the most edges are of size two or three, see Section ??. For our further research, we call the IC-hypergraph from the apdaptec1 netlist H_{A1} . The next section introduces the coarsening approaches from [**?**], the partitioning by an area balanced FM algorithm, and a simple greedy improvement.

4.2 Balanced multi-way hypergraph partitioning

In this section, the partitioning approach that will be introduced is an area balanced hypergraph partitioning approach with balance factor $\beta \in [0, 0.5]$. Our aim is to partition H feasibly into $k > 1$ blocks. A partitioning $P = \{B_1, \ldots, B_k\}$ of H is called *feasible* if the balance restriction

$$\beta A \leq \mu(B_i) \leq (1 - \beta)A \tag{4.2}$$

holds for all $i \in [k]$ where A is the total area of the IC. The next three subsections introduce the three steps coarsening, partitioning or improvement, and refinement.

4.2.1 Coarsening and refinement[5]

Three coarsening approaches are introduced in [**?**]. Each of this approaches use the fusion of vertices or the contraction of an edge[6]. But the refinement is always the same because it is the inversion of the coarsening by maintaining the block numbers. Example ?? introduces the needed steps for the refinement. For detail, let v be the vertex that is created by a fusion of a set of vertices U or by the contraction of an edge e in the coarsening step that leads from H'' to a hypergraph H. Further, let the set W be the set of the vertices that are fused or that belong to e, and let F be the multi-set of all edges of H'' that contain a vertex of W. The steps to do for the

[5] See: [**?**]
[6] See: ??

refinement from a coarser hypergraph H to a finer hypergraph H' are the following. First, we delete all edges that are incident to v in H. Then we delete v in H. Next, we insert all vertices of W to H. Then we insert all edges of F into the current object to get H'. Finally, we delete v from the block of the partition where v belongs to and insert there all vertices of W. We see that the refine step needs W and F. For that, we introduce a map R that assigns to each vertex a pair that comprises a set and a multi-set. So, if we fuse a set of vertices or contract an edge e in H'' to a coarser vertex v then we add v to R and assign to it the pair of the set W of the vertices that are fused or belong to e and the multi-set F of all edges of H'' that contain a vertex of W. Now we introduce the coarsening approaches of Karypis et al. Let $H = (V, E)$ be a hypergraph without loops that we want to coarsen. The first approach is called *edge coarsening* (EC). For this we need the following weights between any two vertices $u, v \in V$

$$w_V(u, v) = \sum_{e \in F} w_E(e)$$

with $F = \{f \in E \mid f \cap \{u, v\} \neq \emptyset\}$ and

$$w_E(e) = \begin{cases} \sum_{i=1}^{k} w_E(e_i) & \text{, If } e_1, \ldots, e_k \text{ collapsed to } e \text{ by an earlier coarsening.} \\ \frac{1}{|e|-1} & \text{, If } |e| > 1. \\ 0 & \text{, Otherwise.} \end{cases}$$

for all $e \in F$. Then, we do the following steps for each coarsening step of the EC. First, we select uniformly at random a vertex $v \in V(H')$ that was not selected before in this coarsening step where H' is the hypergraph of this coarsening step. Next, we fuse v with a vertex $u \in N(v)$ that was not selected before in this coarsening step and achieve maximal weight $w_V(v, u)$ with v. If the fusion of v and u creates a loop then this loop will be deleted.

Remark:
This approach interprets hyperedges as an adjacency relation. Thus, a hyperedge could be replaced by a clique of the vertices of this hyperedge. But this replacement does not respect the IC structure, because three or more elements of an IC do not form a clique in this IC.

Figure ?? shows schematically the effect of the approach EC.

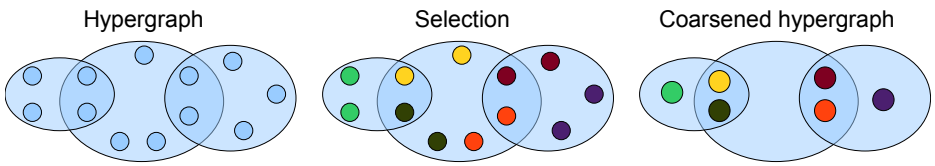


Figure 4.5: The effect of EC schematically.

The second approach is called *hyperedge coarsening* (HEC). The disadvantages of the first approach are that this approach decreases the number of vertices and the number of edges very slowly because only pairs of vertices are selected and so only edges of size two can be

removed. We want to achieve a stronger decrease of the number of vertices and edges. Thus, the HEC approach contracts any edge of an independent set of edges. To get this set, we first sort the edges of the hypergraph of this coarsening step by their edge-weight in a non-increasing order. Next, we sort the edges of the same weight in a non-decreasing edge size order. Then, we iterate the edges with respect to their arrangement and contract each edge if this edge does not contain a, in this coarsening step, fused vertex. Figure ?? shows schematically the effect of the approach HEC.

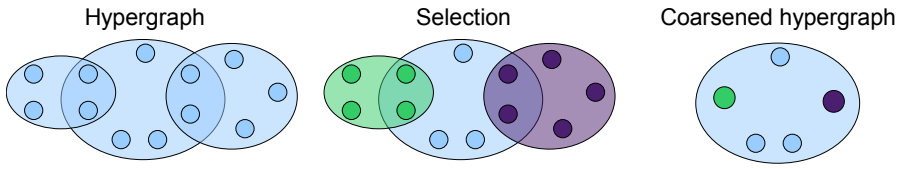


Figure 4.6: The effect of HEC schematically.

The last approach is called *modified hyperedge coarsening* (MHEC). We see in Figure ?? that three vertices in the big edge are untouched during the HEC coarsening step. It is because the big one is the last edge that is visited and at that time it has fused vertices. This leads to the following two disadvantages:

(i) The size of many edges does not decrease sufficiently.

(ii) In the coarser hypergraphs there are some vertices which have not been fused with any other, and some vertices which have been selected for fusion several times. This distorts the shape of the hypergraph.

The MHEC approach is the modification of the HEC approach by fusing the untouched vertices after each coarsening step. So, we perform the HEC approach at first, and then we iterate the edge list again and fuse all vertices in the edges that were not fused before in this coarsening step. If an edge has only one vertex that was not fused before in this coarsening step then we let this vertex untouched. Figure ?? shows schematically the effect of the approach MHEC.

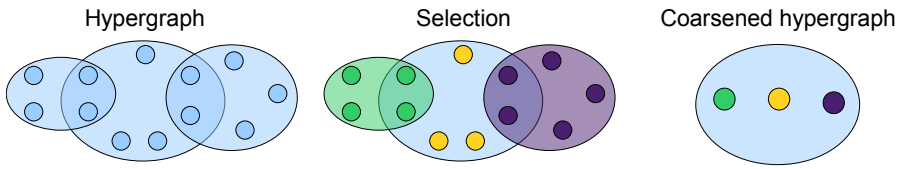


Figure 4.7: The effect of MHEC schematically.

We observe that the MHEC approach reduces the number of vertices and the number of edges faster than the other approaches, and thus we use it for the coarsening.

4.2.2 Partitioning by the area balanced FM algorithm[7]

Karypis et al. use the balanced FM algorithm for the partitioning[8]. This bipartitioning algorithm from Fiduccia and Mattheyses, FM algorithm for short, is an iterative minimum cut heuristic for hypergraph bipartitioning, where this algorithm iterates a given partition by small local changes. We first introduce this balanced bipartition algorithm and then we introduce two generalisations where the last one leads to a multi-way partitioning algorithm. The given partition $\{B_1, B_2\}$ that the FM algorithm needs will be achieved by a random initialisation of two blocks. For this, we iterate the vertex set of the given hypergraph and choose a block number b uniformly at random from $\{1, 2\}$ for each vertex. Then the block number b will be assign to the vertex v if $B \cup \{v\}$ satisfies $(??)$ where B is the current block with block number b, otherwise we insert v into the other block. If we have a partition then the FM algorithm reduces the cut of this partition by appropriate shifts of vertices from their current block to their complementary block. For this, we calculate the number of edges that the cut decreases if we shift a vertex from block $fromBlock$ to its complementary block $toBlock$ for each vertex of the vertex set. We call this number the *gain* $g(v, fromBlock, toBlock)$ of a vertex v. We write $g(v)$ for short if it is clear which blocks are meant. Note that the gain could also be negative. So, if C is the cut before the FM algorithm is applied then

$$C_{new} = C - \sum_{v \in S} g(v)$$

is the new cut where S is the set of vertices that are shifted during the FM algorithm. Further, we want to avoid infinite loops of vertex shifts between the two blocks so we shift each vertex just once. Therefore, we call a vertex *locked* if it is shifted and otherwise we call it *free*. So the main part of the FM algorithm is to shift the free vertex of the highest gain. After that, the algorithm locks the vertex. Now, we focus on the gain calculation. There are three cases how an edge e influences the gain:

(i) One vertex v of e is in the complementary block than the other vertices in e. So the gain of v respective to e is one. The gain of all the other vertices respective to e is zero.

(ii) The number of vertices of both blocks in e is greater than one. So the gain of all vertices respective to e is zero.

(iii) All vertices of e lies in one block. So the gain of all vertices respective to e is minus one.

Figure ?? shows this three cases schematically. In this figure, the two blocks are presented by the colours red and green and the vertices are denoted with the gain of each vertex respective to the edge.

[7] See: [**?**]
[8] See: [**?**]

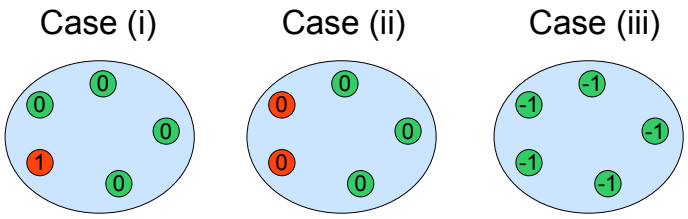


Figure 4.8: The three cases how an edge influences the gain.

Let $H = (V, E)$ be a partitioned hypergraph with partition $\{B, B'\}$ so we have

$$g(v) = \begin{cases} \sum_{e \in E} [v \in e \text{ and } (e \cap B = \{v\} \text{ or } e \cap B' = \{v\})] \\ \qquad - [v \in e \text{ and } (e \cap B = \emptyset \text{ or } e \cap B' = \emptyset)] & \text{, if } v \text{ is free.} \\ -\infty & \text{, otherwise.} \end{cases}$$

for all $v \in V$. We observe that $g(v) \in [-\Delta_H, \Delta_H]$ holds if $v \in V$ is free. Further, we do not have to iterate over the whole edge set. It is enough to iterate over the set of edges that contain the vertex. So we have:

$$g(v) = \begin{cases} \sum_{e \in \{f \in E | v \in f\}} [(e \cap B = \{v\} \text{ or } e \cap B' = \{v\})] \\ \qquad - [(e \cap B = \emptyset \text{ or } e \cap B' = \emptyset)] & \text{, if } v \text{ is free.} \\ -\infty & \text{, otherwise.} \end{cases}$$

So, the time to calculate $g(v)$ decreases from $\mathcal{O}(|E|)$ to $\mathcal{O}(\Delta_H)$ if we get the set of edges that contain v in $O(1)$[9]. The FM algorithm uses this to determine which vertex should be shifted next. For this, we create a list L of $2\Delta_H + 1$ arrays. Each array consists of all free vertices of the same gain, and the arrays of L are sorted by the gain of there elements in ascending order. So, the algorithm calculates the gain for any vertex v and inserts them in the correct array of L. Then, the algorithm chooses the highest index $i \in [-\Delta_H, \Delta_H]$ for which the list L has a nonempty array and shifts the first vertex of this array. After each shift of a vertex v the algorithm locks v, remove v from L, and updates the list L for the neighbours of v. So, the algorithm updates the gain value g_w for any $w \in N(v)$ and moves w into the correct array in L. So, let v be the vertex that should be shifted w.l.o.g. from block B into block B'. Further let e be an edge that contains v. We have the following four cases for the update of the gain of the free vertices in e:

(i) If $|e \cap B'| = 0$ before we shift v then we update the gain as follows $g_w = g_w + 1$ for all $w \in e$ that are free.

(ii) If $|e \cap B'| = 1$ before we shift v then we update the gain as follows $g_w = g_w - 1$ for all $w \in (e \cap B')$ that are free.

(iii) If $|e \cap B| = 0$ after we shift v then we update the gain as follows $g_w = g_w - 1$ for all $w \in e$ that are free.

[9] This could be achieved by an appropriate data structure like the bipartite representation of a hypergraph.

(iv) If $|e \cap B| = 1$ after we shift v then we update the gain as follows $g_w = g_w + 1$ for all $w \in (e \cap B)$ that are free.

Figure ?? shows this four cases schematically. In this figure, block B is coloured green and block B' is coloured red. The vertices are denoted with the gain of each vertex.

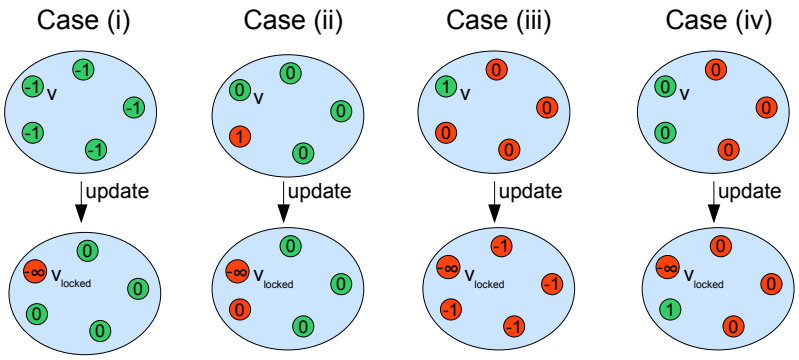


Figure 4.9: The four cases how an edge influences the update of the gain list.

Algorithm ?? presents the steps.

Algorithmus 4: FM

Input: A finite hypergraph H.

Output: The partition $\{B, B'\}$ of H.

1: Initialise two blocks B and B' randomly.
2: Calculate the gain of each vertex in B and B'.
3: Save all vertices of the same gain into an array and save the arrays sorted by the gain of the first vertex of this arrays in ascending order into a list L.
4: $i = \Delta_H$
5: **while** $i \geq -\Delta_H$ **do**
6: $Mg = 0$
7: **for** $v \in L(i)$ **do**
8: Shift v to the other block.
9: Lock v.
10: Delete v from $L(i)$
11: Update the gain and the list L for the neighbours of v.
12: **if** The maximum gain of a neighbour w of v is greater than Mg **then**
13: $Mg = g(w)$
14: **end if**
15: **end for**
16: **if** $i < Mg$ **then**
17: $i = Mg + 1$
18: **end if**
19: $i = i - 1$
20: **end while**
21: **return** $\{B, B'\}$

The worst case computation time, per vertex shift, of the FM algorithm grows linear with the size of the hypergraph[10]. It follows that the Algorithm **??** has a runtime of $\mathcal{O}(|E(H)|)$ because we calculate the gain successively by iterating the edge set $E(H)$, determining the gain of each vertex of the current edge with respect to this edge, and adding this gain to the current gain values of this vertices. A generalisation of the FM algorithm was suggest by Krishnamurthy in [**?**], where he generalized the gain concept. So, let $H = (V, E)$ be a partitioned hypergraph with partition $P = \{B, B'\}$ then the *i-th level gain* of a vertex $v \in V$ is defined by

$$g_i(v) = \sum_{e \in \{f \in E | v \in f\}} [\chi_B(e) = i \text{ and } \chi_{B'}(e) > 0] - [\chi_B(e) > 0 \text{ and } \chi_{B'}(e) = i - 1]$$

[10] See: [**?**]

with $i \in \mathbb{N}$, $v \in B$ and $\chi_p(e) = \begin{cases} |p \cap e| & \text{, if } |\{c \in e \mid c \in p, c \text{ is locked}\}| = 0 \\ \infty & \text{, if } |\{c \in e \mid c \in p, c \text{ is locked}\}| > 0 \end{cases}$ with $p \in P$. We have $g(v) = g_1(v)$ if all vertices are free. If we use l levels then we collect all l level gains in a vector, called *gain vector*, $\Gamma_l(v) = (g_1(v), \ldots, g_l(v))$ for all $v \in V$. All gain vectors are sorted lexicographically, and we choose the vertex with the largest gain vector for the next shift. Krishnamurthy shows in [**?**] that the runtime of this approach is $\mathcal{O}(l|E|)$. We choose $l = 1$ to achieve a minimal runtime, because we want to deal with large hypergraphs. A strategy to get more than two blocks is to use the FM algorithm repeatedly. So, we achieve 2^k blocks if we perform k partitioning steps where we applying the FM algorithm to all current blocks. The algorithm starts with partition $P = \{V(H)\}$, and splits the current block into two if the two split blocks satisfy the balance criteria

$$\beta' \mu(B) \leq \mu(B') \leq (1 - \beta')\mu(B) \tag{4.3}$$

where B is the current block, $B' \in \{\text{split blocks from } B\}$, and $\beta' \in [0, 0.5]$. That is, if a split of a block will create two blocks, where one of them violates (**??**), then we do not split this block. We use this approach for further research and call it FM algorithm, too. Also, we have $\beta = \frac{\min_{B \in P}(\beta' \mu(B))}{A}$. We use this balance criteria because β' is easier to choose than β. Note that a more general FM approach based on the idea of Krishnamurthy, but with respect to all k blocks instead of using the FM algorithm repeatedly, has been introduced by Sanchis in [**?**]. Before we go ahead, we have to decide which value should get the balance factor β'. Table **??** shows two runs of the FM algorithm for H_{A1} with $\beta' = 0.25$ and $\beta' = 0.45$.

Number of vertices after coarsening	β'	Number of blocks	Cut ratio
3778	0.25	32	$\frac{27727}{221142} \approx 0.1254$
3797	0.45	32	$\frac{33318}{221142} \approx 0.1507$

Table 4.1: Cut of 32 blocks with $\beta' = 0.25$ and $\beta' = 0.45$.

We see that a lower balance factor leads to a lower cut. Figure **??** shows the partition profile of the results of this two runs with respect to the number of vertices where the blocks are sorted by the number of vertices.

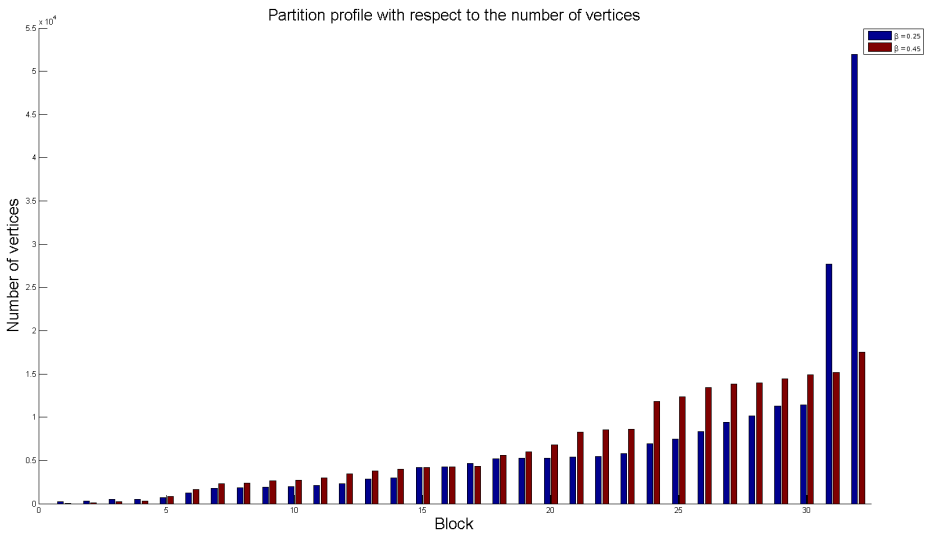


Figure 4.10: Partition profile of the 32 partitions create by the FM algorithm with $\beta' = 0.25$ and $\beta' = 0.45$ with respect to the number of vertices per block.

Figure ?? shows the partition profile of the results of this two runs with respect to the area consumption of the vertices of the blocks where the blocks are sorted by the area consumption of their vertices.

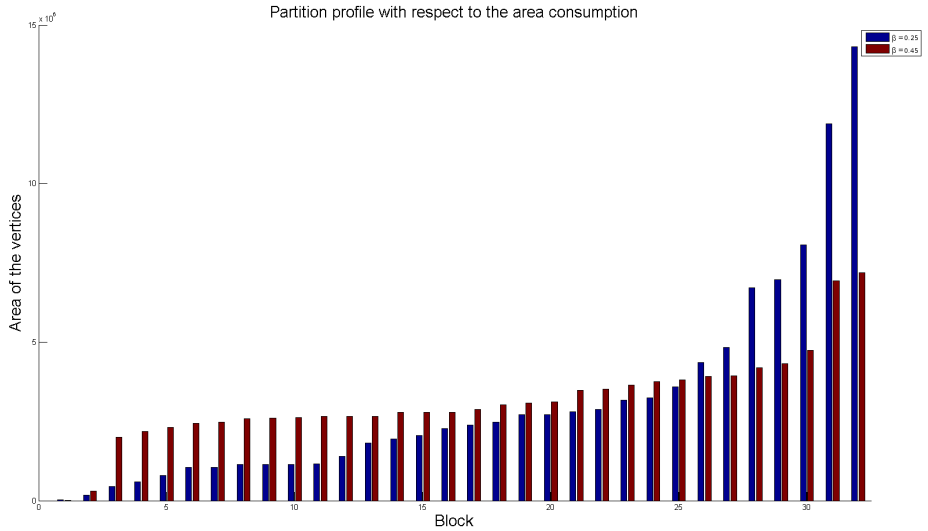


Figure 4.11: Partition profile of the 32 partitions created by the FM algorithm with $\beta' = 0.25$ and $\beta' = 0.45$ with respect to the area consumption of vertices per block.

We see that the FM algorithm, with $\beta' = 0.25$, creates blocks with many vertices and high area consumption. But a block with many vertices and high area consumption is unfavourable for the top-down strategy because in the second step of this strategy we place and wire the elements of the blocks, and doing this with many vertices is too time-consuming. So, we choose $\beta' = 0.45$ for further research. Next, we choose an appropriate number of blocks. Table ?? shows some results with different numbers of blocks.

Number of blocks	Cut ratio	Dependency ratio
8	$\frac{23885}{221142} \approx 0.1080$	$\frac{28}{28} = 1.0000$
16	$\frac{27861}{221142} \approx 0.1260$	$\frac{116}{120} \approx 0.9667$
32	$\frac{33890}{221142} \approx 0.1532$	$\frac{483}{496} \approx 0.9738$
63	$\frac{37818}{221142} \approx 0.1710$	$\frac{1743}{1953} \approx 0.8925$
115	$\frac{41599}{221142} \approx 0.1881$	$\frac{4828}{6555} \approx 0.7365$
214	$\frac{49910}{221142} \approx 0.2257$	$\frac{12886}{22791} \approx 0.5654$

Table 4.2: The FM algorithm with different numbers of blocks.

We see that if we increase the number of blocks than the number of dependencies decreases and the number of cuts increases. The decreasing number of dependencies provides the communication and logic parts model of an IC-hypergraph, because some parts are not connected and so, if this parts belong to different blocks, then the corresponding vertices of the block connectivity graph are not connected, too. Further, the random initialisation of the blocks could lead to high cuts. To reduce the influence of the initialisation, we can restart the FM algorithm several times. Also, we can perform several repetitions of the algorithm to achieve a better cut of the current partition. We call this two parameters $FMrest$ and $FMrep$. Table ?? presents the effect for 32 blocks.

Number of blocks	$FMrest$	$FMrep$	Cut ratio	Dependency ratio	Required time
32	1	1	$\frac{33890}{221142} \approx 0.1532$	$\frac{483}{496} \approx 0.9738$	6 min.
32	1	2	$\frac{32407}{221142} \approx 0.1465$	$\frac{476}{496} \approx 0.9597$	6 min.
31	1	5	$\frac{33665}{221142} \approx 0.1522$	$\frac{435}{465} \approx \mathbf{0.9355}$	5 min.
31	1	10	$\frac{33200}{221142} \approx 0.1501$	$\frac{442}{465} \approx 0.9505$	5 min.
32	2	1	$\frac{31804}{221142} \approx 0.1438$	$\frac{486}{496} \approx 0.9798$	7 min.
32	5	1	$\frac{30324}{221142} \approx 0.1371$	$\frac{466}{496} \approx 0.9395$	8 min.
29	10	1	$\frac{29496}{221142} \approx 0.1334$	$\frac{392}{406} \approx 0.9655$	11 min.
29	10	10	$\frac{29195}{221142} \approx \mathbf{0.1320}$	$\frac{399}{406} \approx 0.9828$	18 min.

Table 4.3: The FM algorithm with different $FMrest$ and $FMrep$ settings.

We see that the cut gets better if we increase the number of restarts or repetitions, but the required time also increases. The minimal dependency ratio was achieved with 5 repetitions and one restart. The best cut was achieved with 10 repetitions and 10 restarts. The results provide the fact that the cut and dependency ratio depend on the random initialisation of the blocks by the FM algorithm. The next section introduces an improvement strategy to get a better cut.

4.2.3 Improvement

After the partitioning of the coarser hypergraph we refine it step by step to get a partitioning of the initial hypergraph. So, after each refine step we get a hypergraph with more vertices and so with more possibilities for the partitioning. To improve the finer hypergraph means to use this possibilities. For that, we introduce a greedy strategy that shifts a vertex to the block where the minimum cut decreases the most. We shift each vertex just once to avoid infinite loops of shifts as in the FM algorithm. Further, it is clear that we only shift a vertex if it has a neighbour that belongs to a different block. So, we iterate the set of vertices in random fashion. If a vertex v has a neighbour that belongs to a different block than v then we determine the block with the highest gain for this vertex and shift v to this block. Algorithm ??, which is called *Greedy-Improve algorithm*, *Gr* for short, presents the steps.

Algorithmus 5: Greegy-Improve

Input: A finite partitioned hypergraph $H = (V, E)$ with partition P.

Output: The improved partition P' of H.

```
1:  for v ∈ V do
2:      maxGain = 0
3:      block = ∅
4:      Initialise blockOfV with the block of v.
5:      for B ∈ P do
6:          if N(v) ∩ B ≠ ∅ and v ∉ B then
7:              gain = g(v, blockOfV, B)
8:              if gain > maxGain then
9:                  maxGain = gain
10:                 block = B
11:             end if
12:         end if
13:     end for
14:     if blockOfV ≠ block then
15:         Remove v from blockOfV.
16:         Insert v to block.
17:     end if
18: end for
19: return P
```

The runtime of Algorithm ?? is $\mathcal{O}(|V| \cdot |P| \cdot (r_H + \Delta_H))$ because we have $|V|$ iterations of the first loop and $|P|$ iterations of the second loop where in each iteration we determine the neighbours

of a vertex which needs r_H[11] steps in worst case and we calculate the gain which needs Δ_H steps in worst case. While $|P| \ll |V(H)|$ we have $\mathcal{O}(|V(H)| \cdot (r_H + \Delta_H))$ for the runtime, but this runtime is a rough approximation because the determining of the neighbours and also the gain calculation need fewer steps than r_H and Δ_H for the most vertices. Table ?? presents the effect of the greedy improvement after a FM partitioning of a MHEC coarsening, for short, the effect of the FM-Gr strategy.

Name	Number of blocks	$FMrest$	$FMrep$	Cut ratio	Dependency ratio	Required time
FM-Gr	31	10	10	$\frac{25788}{221142} \approx 0.1166$	$\frac{435}{465} \approx 0.9355$	129 min.

Table 4.4: The effect of the FM-Gr strategy.

Further, Figure ?? shows the time consumption of the Greedy-Improve algorithm per step, where we coarsen H_{A1} six times.

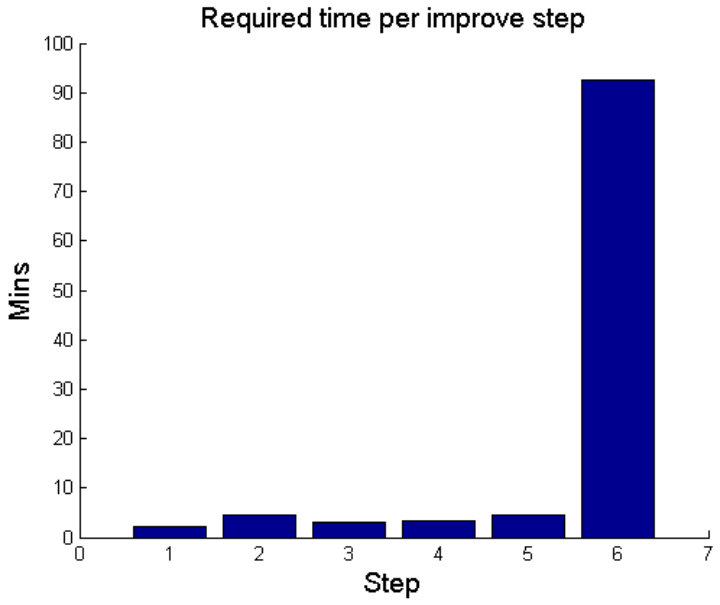


Figure 4.12: Time consumption of the Greedy-Improve algorithm for the H_{A1} per step.

We see that in the last step, where the number of vertices increases from 98297 to 211447 and the number of edges increases from 157806 to 221142, the time consumption increases strongly. This could be because a lot of edges and vertices are inserted at the last refinement step. So, we apply the improve algorithm to the current hypergraph if its number of vertices is at most 10^5. We call this algorithm *limited Greedy-Improve algorithm* or just *LGr* for short. Table ?? presents the effect of this algorithm for a FM partitioning and a MHEC coarsening, for short, the effect of the FM-LGr strategy.

[11] If we use the bipartite representation as data structure.

Name	Number of blocks	$FMrest$	$FMrep$	Cut ratio	Dependency ratio	Required time
FM-Gr	31	10	10	$\frac{25788}{221142} \approx 0.1166$	$\frac{435}{465} \approx 0.9355$	129 min.
FM-LGr	31	10	10	$\frac{26361}{221142} \approx 0.1192$	$\frac{434}{465} \approx 0.9333$	35 min.

Table 4.5: FM-Gr and FM-LGr strategy in comparison.

We see that the required time of the vertex count limited Greedy-Improve algorithm is much lower than in the original case and the difference in cut are just 573 edges. So we prefer the vertex count limited approach for further research. Further, we can repeat the improve algorithm several times. We call this parameter $LGRrep$. Table ?? presents the effect with $FMrep = FMrest = 10$.

Name	Number of blocks	$LGRrep$	Cut ratio	Dependency ratio	Required time
FM-Gr	31	1	$\frac{25788}{221142} \approx 0.1166$	$\frac{435}{465} \approx 0.9355$	129 min.
FM-LGr	31	1	$\frac{26361}{221142} \approx 0.1192$	$\frac{434}{465} \approx \mathbf{0.9333}$	35 min.
FM-LGr	32	2	$\frac{25489}{221142} \approx \mathbf{0.1153}$	$\frac{470}{496} \approx 0.9476$	42 min.
FM-LGr	32	5	$\frac{27183}{221142} \approx 0.1229$	$\frac{467}{496} \approx 0.9415$	67 min.

Table 4.6: The limited Greedy-Improve algorithm with several repetitions.

We see in table ?? that the cut that is achieved with five repetitions is higher than the cut with two repetitions. That is because the cut depends on the random initialisation of the blocks by the FM algorithm.

4.3 Random walk coarsening

The first thing you can modify in the hierarchical partitioning process is the coarsening step. It seems that the MHEC does not respect the structure of the IC-hypergraph. That's because the MHEC approach only sorts the edges appropriately and contracts them. We want to consider the structure a little more. So, we fuse all vertices of a path of the IC-hypergraph instead of contracting single edges, where this path is created by a random walk on the dual of the IC-hypergraph[12]. We modify the random walk in this manner that we lock a visited vertex in H^* and forbid it to visit locked vertices again. So, the output of the random walk is a path. Further, this path is a sequence of vertices in H^* and so it is a sequence of edges in H. Next, we fuse all vertices of all edges of this sequence of edges. Then we lock all vertices in H^* that correspond

[12] See Section ??

to an edge in H that contain the new vertex. Then we go on with the coarsening until all vertices in H^* are locked. It is clear that if we run the random walk for a long time then we select a lot of edges in H. It follows that the fusion of all vertices of this edges creates a big vertex of high degree. So, we only run the random walk for $l' \in \mathbb{N}$ steps on H^*. We call the number of steps the *length of the random walk*. Further, we can select $w \in \mathbb{N}$ vertices in H^* per step instead of one if it is possible. "If it is possible" means that the current vertex in H^* has at least w unlocked neighbours. Otherwise, we select as many vertices as possible. We call w the *width of the random walk*. The steps of the *RW-Coarsen algorithm*, *RW-Coarsen* for short, are presented below.

Algorithmus 6: RW-Corasen

Input: A finite hypergraph H, the random walk length l' and the random walk width w.

Output: The coarser hypergraph H'.

1: **while** There is a non locked vertex in H^* **do**
2: Select a start vertex e uniformly at random from the unlocked vertices of H^*.
3: Perform a random walk of length l' and width w on the unlocked vertices H^* that starts at e and saves all vertices of the in H visited edges that are selected by the random walk on H^* into a list L.
4: Fuse all vertices of L in H.
5: Lock all vertices in H^* that correspond to an edge in H that contains the fused vertex.
6: **end while**
7: **return** H

The runtime is $\mathcal{O}(E(H))$. Table **??** presents the effect of different lengths and widths of the RW-Coarsen algorithm.

Name	Length	Width	Number of vertices after 20 runs	Number of edges after 20 runs	Required time
MHEC	-	-	391	1167	3.12 min.
RW-Coarsen	1	1	22180	69866	1.91 min.
RW-Coarsen	5	1	22499	69559	2.08 min.
RW-Coarsen	10	1	22309	69233	1.84 min.
RW-Coarsen	100	1	22431	70287	1.76 min.
RW-Coarsen	100	2	22423	70257	1.79 min.
RW-Coarsen	100	5	22061	**64964**	1.82 min.
RW-Coarsen	100	10	22013	68875	1.76 min.
RW-Coarsen	1000	1	**21640**	68201	1.80 min.
RW-Coarsen	1000	2	22115	68558	1.77 min.
RW-Coarsen	1000	5	21641	68876	1.75 min.
RW-Coarsen	1000	10	22041	70700	1.75 min.

Table 4.7: Comparison of the two approaches MHEC and RW-Coarsen.

We see in table **??** that the MHEC approach reduces the vertices and edges more than the RW-Coarsen approach. The RW-Coarsen approach achieved its best result with respect to the number of vertices with length 1000 and width 1. But this result is just a little better than the others and it is bad with respect to the MHEC approach. Further, we observe that it is unfavourable to contract or to fuse the vertices of a big edge. Because, big edges could be refresh wires or clock wires and so they connect different parts of the IC. So, if we fuse its vertices we destroy the structure of the IC-hypergraph. Also the fusion of the vertices of a big edge creates a big vertex with high degree and so a lot of vertices in H^* will be locked. We can modify the random walk on H^* in such a way that we just allow a transition between two vertices in H^* if their edges in H are smaller than $l \in \mathbb{N}$. We call the modified algorithm *limited RW-Coarsen* or *LRW-Coarsen* for short. Table **??** presents the effect of this algorithm with length 1000 and width 1.

Name	Runs	l	Number of vertices of the coarser hypergraph	Number of edges of the coarser hypergraph	Required time
MHEC	20	-	391	1167	3.12 min.
RW-Coarsen	20	-	21640	68201	1.80 min.
LRW-Coarsen	13	3	50453	73809	2.25 min.
LRW-Coarsen	20	5	22926	**55974**	1.85 min.
LRW-Coarsen	20	10	**19010**	60367	1.44 min.

Table 4.8: Comparison of the two approaches MHEC and the limited RW-Coarsen.

We see that in the third row only 13 runs were performed. It is because the number of edges, which could be fused, decreases strongly with $l = 3$. So, we see that this modification leads to better results but they are also not good enough for our further research. We have the following disadvantages of the (L)RW-Coarsen approach versus the MHEC approach:

(i) The RW-Coarsen approach could destroy the structure of the IC-hypergraph.

(ii) The MHEC approach uses the hypergraph structure better than the (L)RW-Coarsen approach, because the edge contraction is commutative and so it is not that important how many edges are selected per step by the (L)RW-Coarsen approach if a sufficient number of steps is done.

(iii) The results of the (L)RW-Coarsen approach are not useful for a fast partitioning.

4.4 Bounded modified hyperedge coarsening (BMHEC) and the vertex to block approach

In Section ?? we see that the MHEC approach decreases the number of vertices and the number of edges fast. Thus, we can coarsen the hypergraph as far as each vertex can be interpreted as a block[13]. We call this approach vertex to block approach or VTB for short. Table ?? presents the effect.

Name	Number of blocks	Cut ratio	Dependency ratio	Required time
Parameter: $FMrest = 0$, $FMrep = 0$				
FM	214	$\frac{49910}{221142} \approx 0.2257$	$\frac{12886}{22791} \approx 0.5654$	8 min.
VTB	210	$\frac{1349}{221142} \approx 0.0061$	$\frac{209}{21945} \approx 0.0095$	6 min.

Table 4.9: Comparison of the two approaches FM and the VTB.

The results of the VTB approach are so much better so that something must have gone wrong. So, we look at the partition profile of the VTB approach, shown in Figure ?? where we used a logarithmic scale for the y-axis.

[13] This approach is motivated by the approach of the randomized min-cut algorithm, see [**?**] page 12 et seqq.

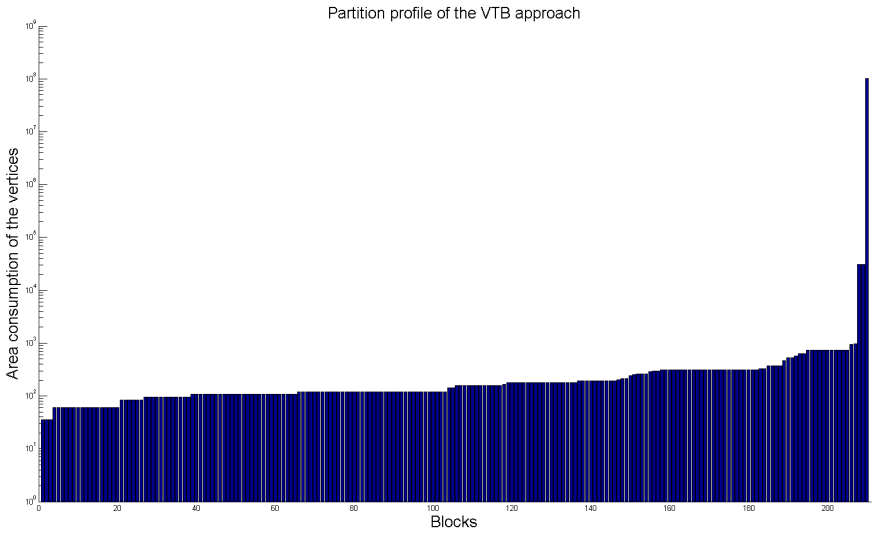


Figure 4.13: Partition profile of the VTB approach.

The blocks are ordered by their area consumption. We see that the last block stands out from all others. It consumes much more area than the other blocks. It is because it contains nearly all vertices namely exactly 211151 of the 211447 vertices of H_{A1}. So, we see the MHEC coarsening is not a good choice for the VTB approach. We introduce a new coarsening approach that is motivated by the result of the LRW-Coarsen approach and that bases on the MHEC approach. We call it bounded modified hyperedge coarsening, BMHEC for short. The BMHEC approach modifies the MHEC approach in such a way that an edge is contracted if the resulting vertex v is bounded above by

$$\mu(\{v\}) \leq \gamma A$$

where A is the total area of the IC and $\gamma \in [0, 1]$ is a factor for the upper bound. Note that, if we choose γ too large the behaviour of the BMHEC approach is the same as the behaviour of the MHEC approach. We compare this two approaches in table **??**.

Name	Runs	γ	Number of vertices of the coarser hypergraph	Number of edges of the coarser hypergraph	Required time
MHEC	20	-	391	1167	3.12 min.
BMHEC	20	0.5	348	**33825**	2.59 min.
BMHEC	20	0.2	349	49286	1.13 min.
BMHEC	20	0.1	341	48348	1.08 min.
BMHEC	20	0.05	**262**	52890	0.91 min.
BMHEC	20	0.01	513	63002	0.85 min.
BMHEC	20	0.001	1520	72878	0.95 min.

Table 4.10: Comparison of the two approaches MHEC and the BMHEC.

We observe that the BMHEC approach leads to a higher number of edges than the MHEC approach but the BMHEC approach leads to a lower number of vertices than the MHEC approach if $\gamma \geq 0.05$. It is because the balance criteria forbids some edge contractions. So, it could be that we have an edge e with a low number of incident vertices but we do not contract e if the vertices that belong to e consume too much area. Now we use the BMHEC approach for the VTB partitioning. We call this partitioning VTB'. Table ?? presents the effect.

Name	γ	Number of blocks	Cut ratio	Dependency ratio	Required time
Parameter: $FMrest = 0$, $FMrep = 0$					
FM	-	214	$\frac{49910}{221142} \approx 0.2257$	$\frac{12886}{22791} \approx 0.5654$	8 min.
VTB'	0.1	210	$\frac{47859}{221142} \approx \mathbf{0.2152}$	$\frac{694}{21945} \approx \mathbf{0.0316}$	10 min.
VTB'	0.05	208	$\frac{50728}{221142} \approx 0.2294$	$\frac{865}{21528} \approx 0.0402$	8 min.
VTB'	0.01	319	$\frac{59824}{221142} \approx 0.2705$	$\frac{5843}{50721} \approx 0.1152$	13 min.

Table 4.11: Comparison of the two approaches FM and the VTB'.

In table ?? we see that the VTB' approach with $\gamma = 0.1$ is similar to the result of the FM approach with 214 blocks with respect to the cut ratio, but the VTB' approach has a better dependency ratio. Figure ?? shows the partition profile of the VTB' approach with $\gamma = 0.1$ and with a logarithmic scale for the y-axis.

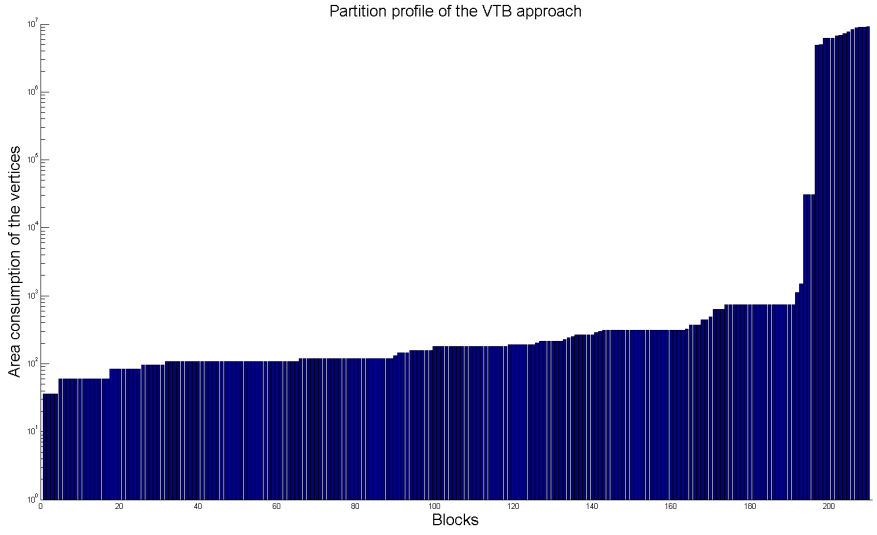


Figure 4.14: Partition profile of the VTB' approach.

In Figure ?? we see that the most blocks have a low area consumption where this blocks do not differ a lot in their area consumption. There also are some blocks with a higher area consumption, but these also do not differ a lot in their area consumption. This behaviour will be used in the next section to get a new partition approach.

4.5 VTB' dependency minimization approach

As we already saw in Section ?? the coarsening strategy BMHEC allows us to interpret each vertex of the coarser hypergraph as a block of a partition. On this basis we focus on the minimization of dependencies in this section. So, we have a coarser hypergraph H_{BMHEC} with partition $\{\{v\} \mid v \in V(H_{BMHEC})\}$. The idea is to unify those two blocks that have most of their neighbours in common. To avoid blocks that consume a lot of area and so consist of many vertices, we perform the union $B \cup B'$ only if the result satisfies

$$\mu(B \cup B') \leq \gamma' A \tag{4.4}$$

where A is the total area of the IC and $\gamma' \in [0, 1]$ is the upper bound factor. It means that we unify this two blocks that have most of their neighbours in common and that the area consumption of the union is less or equal than $\gamma' A$. The steps of the *vertex to block for dependency minimization algorithm*, *VTB'-DM* for short, are presented in Algorithm ??.

Algorithmus 7: VTB'-DM

Input: A finite hypergraph H, γ, γ' and the number of blocks BN that should be achieved.

Output: A partition P of the hypergraph H.

1: Coarsen H with the BMHEC approach to H_{BMHEC}.
2: Initialize P with $\{\{v\} \mid v \in V(H_{BMHEC})\}$.
3: Create the block connectivity graph of H_{BMHEC} with Algorithm ??.
4: **while** $|P| > BN$ and there are two blocks that can be unified **do**
5: Choose the two blocks B, B' that have most of their neighbours in common and whose total area consumption is less or equal than $\gamma' A$.
6: Unify B and B'.
7: Fuse the two corresponding vertices of the block connectivity graph and remove the loop if this two vertices are connected.
8: **end while**
9: **return** P

The runtime of the Algorithm ?? is $\mathcal{O}(|V(H_{BMHEC})|^2)$.

Lemma 4.3:

Let H be a partitioned hypergraph with partition P. Further, let B and B' be blocks of P that should be unified, let v, w their corresponding vertices in the block connectivity graph $BC = \mathrm{BC}(H, P)$, and let dr be the dependency ratio of BC. Then Algorithm ?? have the following optimization behaviour:

(i) If $dr > \frac{|N(v) \cap N(w)| + [v \in N(w)]}{(|V(BC)| - 1)}$ then the unification of B and B' increases the dependency ratio of the block connectivity graph.

(ii) If $dr = \frac{|N(v)\cap N(w)|+[v\in N(w)]}{(|V(BC)|-1)}$ then the unification of B and B' leaves the dependency ratio of the block connectivity graph as it is.

(iii) If $dr < \frac{|N(v)\cap N(w)|+[v\in N(w)]}{(|V(BC)|-1)}$ then the unification of B and B' decreases the dependency ratio of the block connectivity graph.

Proof: We have B and B' that are the blocks that should be unified and v, w that are their corresponding vertices in the block connectivity graph $BC = \mathrm{BC}(H, P)$. Further $dr = \frac{|E(BC)|}{|V(BC)|\cdot(|V(BC)|-1)/2}$ is the dependency ratio of BC. So, if we unify B and B' then the new dependency ratio could be calculated by:

$$dr_{new} = \left(dr - \frac{|N(v)\cap N(w)| + [v\in N(w)]}{|V(BC)|\cdot(|V(BC)|-1)/2}\right)\cdot\frac{|V(BC)|}{(|V(BC)|-2)}$$

The Algorithm ?? has no effect with respect to the dependencies if $dr_{new} = dr$ holds. We have:

$$\begin{aligned} dr &= dr_{new} \\ &= \left(dr - \tfrac{|N(v)\cap N(w)|+[v\in N(w)]}{|V(BC)|\cdot(|V(BC)|-1)/2}\right)\cdot\tfrac{|V(BC)|}{(|V(BC)|-2)} \\ &= \tfrac{dr\cdot|V(BC)|}{(|V(BC)|-2)} - \tfrac{|N(v)\cap N(w)|+[v\in N(w)]}{(|V(BC)|-1)\cdot(|V(BC)|-2)/2} \end{aligned}$$

Further, it follows that

$$dr - \frac{dr\cdot|V(BC)|}{(|V(BC)|-2)} = -\frac{|N(v)\cap N(w)| + [v\in N(w)]}{(|V(BC)|-1)\cdot(|V(BC)|-2)/2}$$

and that

$$dr\cdot(|V(BC)|-2) - dr\cdot|V(BC)| = -\frac{|N(v)\cap N(w)| + [v\in N(w)]}{(|V(BC)|-1)/2}.$$

Finally, we have

$$(-2)\cdot dr = -\frac{|N(v)\cap N(w)| + [v\in N(w)]}{(|V(BC)|-1)/2}$$

and so

$$dr = \frac{|N(v)\cap N(w)| + [v\in N(w)]}{(|V(BC)|-1)}.$$

This proves (ii). Furthermore, it follows that if $dr > \frac{|N(v)\cap N(w)|+[v\in N(w)]}{(|V(BC)|-1)}$ then $dr < dr_{new}$ which shows (i). Analogue, if $dr < \frac{|N(v)\cap N(w)|+[v\in N(w)]}{(|V(BC)|-1)}$ then $dr > dr_{new}$ which shows (iii). □

Corollary 4.4:

Let H be a partitioned hypergraph with partition P. Then the dependency ratio decrease at most if the two blocks $B, B' \in P$, which should be unified and for which holds $(??)$, satisfies

$$|N(v)\cap N(w)| + [v\in N(w)] \to \max$$

where v, w are the vertices of the block connectivity graph that correspond to B, B'.

Table ?? presents the effect of the Algorithm ?? where we use $\gamma = 0.1$ for the BMHEC approach and where we want to achieve 32 blocks.

Name	γ'	Numb. of blocks	Cut ratio	Dependency ratio	$\frac{\max_{B \in P} \mu(B)}{A}$	Required time
Parameter: $FMrest = 1$, $FMrep = 5$						
FM	-	31	$\frac{33665}{221142} \approx 0.1522$	$\frac{435}{465} \approx 0.9355$	0.0684	5 min.
Parameter: $FMrest = 1$, $FMrep = 5$						
FM-Gr	-	31	$\frac{25788}{221142} \approx 0.1166$	$\frac{435}{465} \approx 0.9355$	0.0720	129 min.
VTB'-DM	0.1	56	$\frac{52483}{221142} \approx 0.2373$	$\frac{332}{1540} \approx 0.2156$	0.0552	23 min.
VTB'-DM	0.2	47	$\frac{45318}{221142} \approx 0.2049$	$\frac{127}{1081} \approx 0.1175$	0.1413	21 min.
VTB'-DM	0.3	37	$\frac{51815}{221142} \approx 0.2343$	$\frac{152}{666} \approx 0.2282$	0.0874	22 min.
VTB'-DM	0.5	56	$\frac{38659}{221142} \approx \mathbf{0.1748}$	$\frac{76}{1540} \approx \mathbf{0.0494}$	0.2830	20 min.

Table 4.12: Comparison of the approaches FM, GR-Imp and the VTB'-DM.

We achieved the fewest dependency ratio with $\gamma' = 0.5$. We draw the block connectivity graph to better assess the results, and for that we use a 2D spring embedding algorithm. The spring embedding algorithm is an iterative algorithm that could be used to draw a graph, see [**?**] page 72 et seqq. Further, we draw the vertices of different size to visualize the area consumption. So, the biggest vertex corresponds to the biggest block. The other vertices have a size of $\frac{\mu(B')}{\mu(B)} \cdot s$ where B is the biggest block and s is the size of the vertex that corresponds to B. If $\frac{\mu(B')}{\mu(B)} \cdot s < ms$ then the considered vertex gets size ms where ms is the minimum size we allow for a vertex. Also, we draw the edges of different size to visualize between which blocks run a lot of edges. There are three types of thickness for the lines. The thickest line is used to visualize the maximal number of edges me that run between two different blocks. The little thinner line is used if the number of edges between two different blocks is lower than me and greater than $0.5 \cdot me$. The other lines get the smallest thickness. Figure **??** shows the block connectivity graph of the result of the FM algorithm with one restart and five repetitions.

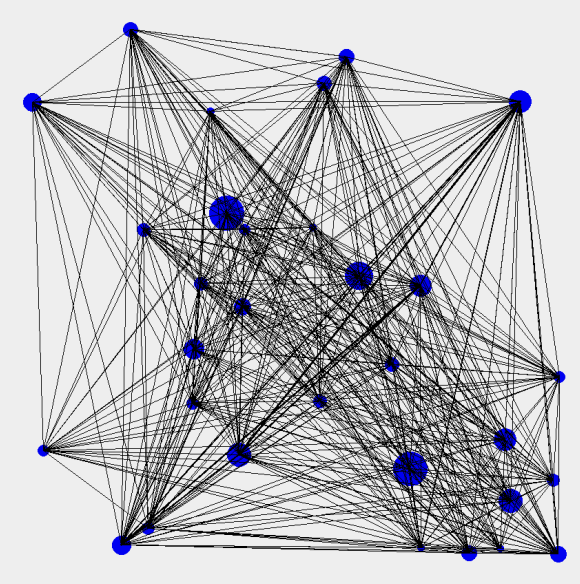


Figure 4.15: Block connectivity graph of the result of the FM algorithm with one restart and five repetitions.

The edge density of 0.9355 implies a lot of edges in the block connectivity graph as we see in Figure ??. Figure ?? shows the block connectivity graph of the result of the VTB'-DM algorithm with $\gamma' = 0.2$.

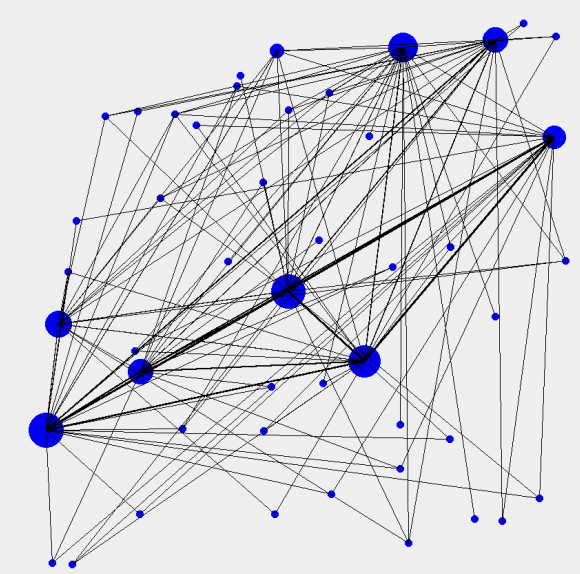


Figure 4.16: Block connectivity graph of the result of the VTB'-DM algorithm with $\gamma' = 0.2$.

For better comparison Figure ?? shows the block connectivity graph of the result of the VTB'-DM algorithm with $\gamma' = 0.5$.

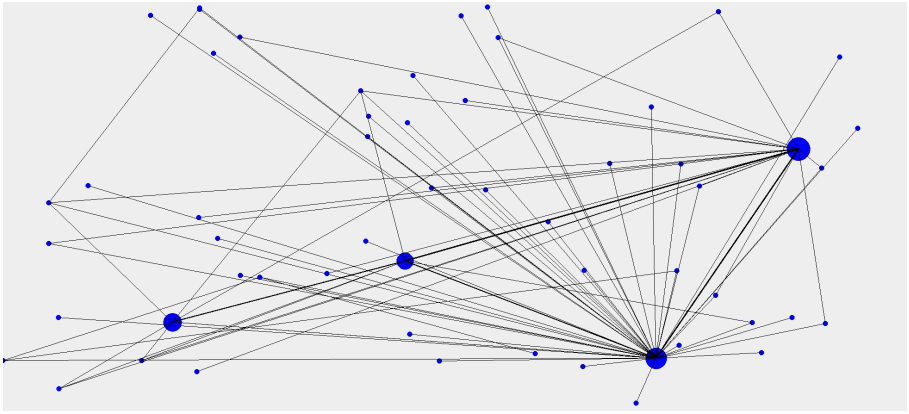


Figure 4.17: Block connectivity graph of the result of the VTB'-DM algorithm with $\gamma' = 0.5$.

We see that the illustrations of the two block connectivity graphs differ strongly as their edge densities, too. But in Figure ??, the vertices of degree less or equal than two disturb. They also do not have to lead to a better placement but they could be interesting for a 3D IC-design. We go on by modifying the VTB'-DM algorithm in this manner that we clean the block connectivity graph from vertices of degree less or equal two. It means that we unify each vertex of degree less or equal two of the block connectivity graph with its neighbour of the lowest area consumption if the resulting block has an area consumption of at most $\gamma' A$ with $\gamma' \in [0, 1]$, whereby A is the area consumption of the IC. This modifications of Algorithm ?? are presented in Algorithm ??.

Algorithmus 8: VTB'-DMC

Input: A finite hypergraph H, γ, γ', and the number of blocks BN that should be achieved.

Output: The partition P of the hypergraph H.

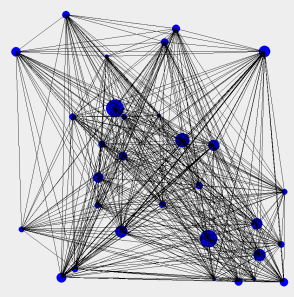
1: Coarsen H with the BMHEC approach to H_{BMHEC}.
2: Initialize P with $\{\{v\} \mid v \in V(H_{BMHEC})\}$.
3: Create the block connectivity graph of H_{BMHEC} with Algorithm ??.
4: **while** $|P| > BN$ and there are two blocks that can be unified **do**
5: Determine this two blocks B and B' that have most of their neighbours in common and whose total area consumption is less or equal than $\gamma' A$.
6: Unify B and B'.
7: Fuse the two corresponding vertices of the block connectivity graph and remove the loop if this two vertices are connected.
8: **end while**
9: **for** $v \in V(BC)$ **do**
10: **if** $deg(v) \leq 2$ and the unification of v and $\arg\min_{w \in N(v)}(\mu(w))$ satisfies $(??)$ **then**
11: Unify v and $\arg\min_{w \in N(v)}(\mu(w))$.
12: **end if**
13: **end for**
14: **return** P

The runtime of the Algorithm ?? is $\mathcal{O}(|V(H_{BMHEC})|^2)$. Table ?? presents the effect of the Algorithm ?? where we use $\gamma = 0.1$ for the BMHEC approach and $\gamma' = 0.2$.

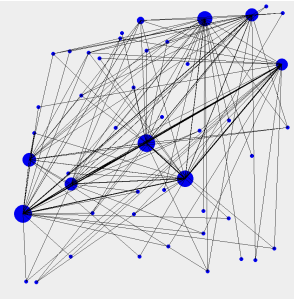
Name	VTB blocks	Req. blocks	Numb. of blocks	Cut ratio	Dependency ratio	Req. time
Parameter: $FMrest = 1$, $FMrep = 5$						
FM	-	-	31	$\frac{33665}{221142} \approx 0.1522$	$\frac{435}{465} \approx 0.9355$	5 min.
Parameter: $FMrest = 10$, $FMrep = 10$						
FM-Gr	-	-	31	$\frac{25788}{221142} \approx 0.1166$	$\frac{435}{465} \approx 0.9355$	129 min.
VTB'-DM	249	32	47	$\frac{45318}{221142} \approx 0.2049$	$\frac{127}{1081} \approx 0.1175$	21 min.
VTB'-DMC	239	32	35	$\frac{51156}{221142} \approx 0.2313$	$\frac{228}{595} \approx 0.3832$	22 min.
VTB'-DMC	344	115	43	$\frac{45647}{221142} \approx \mathbf{0.2064}$	$\frac{206}{903} \approx 0.2281$	21 min.
VTB'-DMC	240	63	46	$\frac{53598}{221142} \approx 0.2423$	$\frac{271}{1035} \approx 0.2618$	21 min.
VTB'-DMC	246	115	80	$\frac{51398}{221142} \approx 0.2324$	$\frac{348}{3160} \approx \mathbf{0.1101}$	21 min.

Table 4.13: Comparison of the four approaches FM, FM-Gr, VTB'-DM and the VTB'-DMC.

Figure ?? shows the three block connectivity graphs to compare them.



Block connectivity graph of the result of the FM algorithm, see Figure ??.



Block connectivity graph of the result of the VTB'-DM algorithm, see Figure ??.

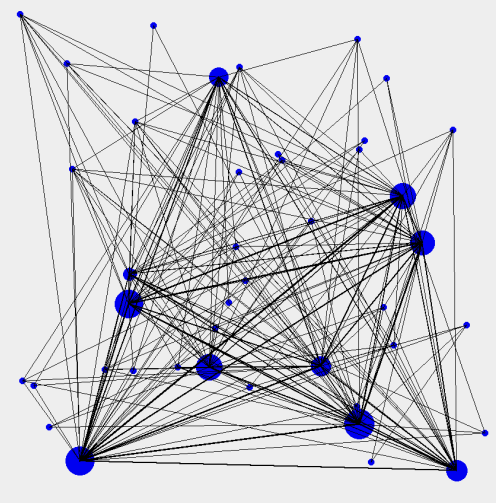


Figure 4.18: Block connectivity graph from Figure ??, Figure ??, and of the result of the VTB'-DMC algorithm with $\gamma' = 0.2$ and 43 blocks.

Figure ?? shows that the cleaning up of the block connectivity graph leads to a graph that has fewer vertices with degree lower or equal two. But there also exist vertices with degree of two or less because the union of two blocks has to satisfy the border restriction $(??)$. Further, the VTB'-DMC algorithm and also the VTB'-DM algorithm lead to a higher cut than the FM algorithm, see table ?? and ??. Another disadvantage is that we have to coarsen the IC-hypergraph strongly, especially those that have more than 10^6 vertices, otherwise the runtime of the Algorithm ?? increases significantly, see Section ??. A strategy to avoid this two disadvantages is to combine the FM algorithm with the VTB'-DMC algorithm. Sometimes, a strategy will be combined with the FM algorithm by a modification of the gain, see [**?**] for example. But this is not an opportunity for the combination of the FM algorithm with the VTB'-DMC algorithm, because the structure of the block connectivity graph is given by the random initialisation of the blocks in the FM algorithm. So, we perform the following strategy. First, we coarsen the hypergraph but not too strong as it is required for the VTB'-DMC approach. This allows us to take a small value for γ, and this implies a gently coarsening by the BMHEC approach. Next, we apply the FM algorithm to the coarser hypergraph to achieve a partition of some blocks. In table ?? we see that if we achieve a partition with enough blocks by the FM algorithm then the number of dependencies decreases.

Then, we apply the DMC algorithm to the output of the FM algorithm. At last, we use the limited greedy improvement to improve the finer hypergraphs until the limitation is reached. We call this strategy *FM-DMC-LGr*. Table ?? presents the results of this strategy and good results of the other approaches for comparison.

Name	Blocks before DM(C)	γ'	Numb. of blocks	Cut ratio	Dependency ratio	Req. time
Parameter: $FMrest = 1$, $FMrep = 5$						
FM	-	-	31	$\frac{33665}{221142} \approx 0.1522$	$\frac{435}{465} \approx 0.9355$	5 min.
Parameter: $FMrest = 10$, $FMrep = 10$						
FM-Gr	-	-	31	$\frac{25788}{221142} \approx 0.1166$	$\frac{435}{465} \approx 0.9355$	129 min.
VTB'-DM	249	0.2	47	$\frac{45318}{221142} \approx 0.2049$	$\frac{127}{1081} \approx 0.1175$	21 min.
VTB'-DMC	344	0.2	43	$\frac{45674}{221142} \approx 0.2064$	$\frac{206}{903} \approx 0.2281$	21 min.
VTB'-DMC	246	0.2	80	$\frac{51398}{221142} \approx 0.2324$	$\frac{348}{3160} \approx 0.1101$	21 min.
Parameter: $FMrest = 10$, $FMrep = 10$, $\gamma = 0.0004$						
FM-DMC-LGr	121	0.24	32	$\frac{34418}{221142} \approx 0.1556$	$\frac{382}{496} \approx \mathbf{0.7702}$	21 min.
FM-DMC-LGr	63	0.24	56	$\frac{29489}{221142} \approx \mathbf{0.1333}$	$\frac{1477}{1540} \approx 0.9591$	24 min.
FM-DMC-LGr	117	0.24	88	$\frac{33353}{221142} \approx 0.1508$	$\frac{3471}{3828} \approx 0.9067$	24 min.

Table 4.14: Comparison of the four approaches FM, FM-Gr, VTB'-DM and the VTB'-DMC.

In Table ?? we see that the FM-DMC-LGr approach does not lead to good results with respect to the dependency ratio. It is because the BMHEC-FM approach leads to block connectivity grpahs with high density and this does not lead to partitions with low dependency ratio.

4.6 Benchmark

Table ?? presents the data of the considered hypergraphs with respect to their netlists.

Netlist	Number of vertices	Number of edges	δ	Δ	Average degree	s	r	Average number of vertices in an edge
adaptec1	211447	221142	1	288	≈ 4.35	1	1270	≈ 4.163
adaptec2	255023	266009	1	371	≈ 4.13	1	1202	≈ 3.979
adaptec3	451650	466758	1	660	≈ 4.09	1	2439	≈ 3.956
adaptec4	496045	515951	1	378	≈ 3.79	1	2833	≈ 3.645
bigblue1	278164	284479	1	378	≈ 4.05	1	1717	≈ 3.967
bigblue2	557866	577235	1	119	≈ 3.77	1	9236	≈ 3.647
bigblue3	1096812	1123170	1	1475	≈ 3.45	1	5106	≈ 3.395

Table 4.15: Data of the hypergraphs with respect to their netlists.

We see:

- The hypergraph H_{A1} with respect to the netlist adaptec1 belongs to $\mathcal{IC}_{2,2}$.
- The hypergraph H_{A2} with respect to the netlist adaptec2, the hypergraph H_{A3} with respect to the netlist adaptec3, the hypergraph H_{B1} with respect to netlist bigblue1, and the hypergraph H_{B4} with respect to netlist bigblue4 belongs to $\mathcal{IC}_{2,1}$.
- The hypergraph H_{A4} with respect to the netlist adaptec4, the hypergraph H_{B2} with respect to netlist bigblue2, and the hypergraph H_{B3} with respect to netlist bigblue3 belongs to $\mathcal{IC}_{1,1}$.

We apply the FM algorithm and the VTB'-DMC approach to this IC-hypergraphs. The following tables present the results of the smaller IC-hypergraphs H_{A1}, H_{A2}, and H_{B1} with their block connectivity graphs.

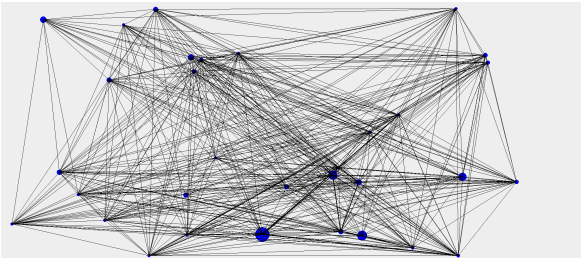
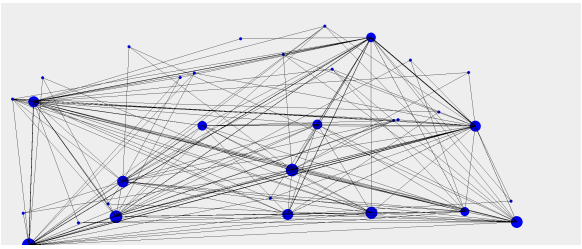
FM with $\gamma = 0.004$, $FMrest = 10$, and $FMrep = 10$		
Number of blocks	32	
Cut ratio	$\frac{28984}{221142} \approx 0.1311$	
Dependency ratio	$\frac{441}{496} \approx 0.8891$	
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.0979	Figure 4.19: $\mathrm{BC}(H_{A1}, P)$ where P is the partition of the FM algorithm.
VTB'-DMC with $\gamma = 0.02$ and $\gamma' = 0.2$		
Number of blocks	32	
Cut ratio	$\frac{51897}{221142} \approx 0.2347$	
Dependency ratio	$\frac{146}{496} \approx 0.2944$	
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.0952	Figure 4.20: $\mathrm{BC}(H_{A1}, P)$ where P is the partition of the VTB'-DMC algorithm.

Table 4.16: The FM algorithm and the VTB'-DMC approach with respect to the H_{A1} hypergraph.

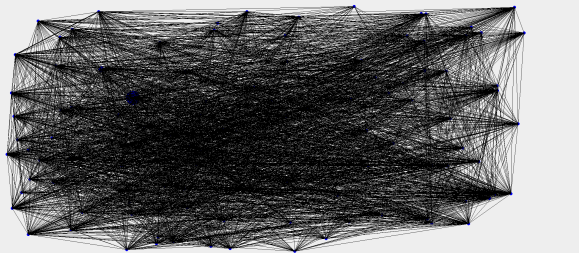
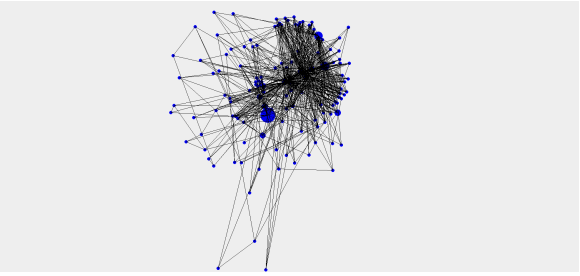
FM with $\gamma = 0.004, FMrest = 10$, and $FMrep = 10$		
Number of blocks	110	
Cut ratio	$\frac{37228}{266009} \approx 0.1340$	
Dependency ratio	$\frac{3158}{5995} \approx 0.5268$	
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.3974	Figure 4.21: $\mathrm{BC}(H_{A2}, P)$ where P is the partition of the FM algorithm.
VTB'-DMC with $\gamma = 0.02$ and $\gamma' = 0.2$		
Number of blocks	130	
Cut ratio	$\frac{66846}{266009} \approx 0.2513$	
Dependency ratio	$\frac{636}{8385} \approx 0.0758$	
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.1568	Figure 4.22: $\mathrm{BC}(H_{A2}, P)$ where P is the partition of the VTB'-DMC algorithm.

Table 4.17: The FM algorithm and the VTB'-DMC approach with respect to the H_{A2} hypergraph.

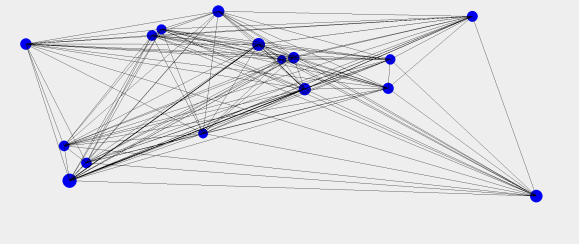
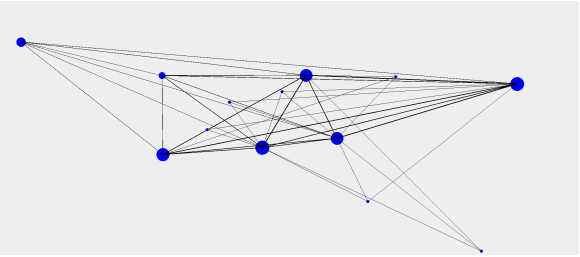
FM with $\gamma = 0.004, FMrest = 10$, and $FMrep = 10$		
Number of blocks	16	
Cut ratio	$\frac{49848}{284479} \approx 0.1752$	
Dependency ratio	$\frac{120}{120} = 1$	
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.0788	Figure 4.23: $\mathrm{BC}(H_{B1}, P)$ where P is the partition of the FM algorithm.
VTB'-DMC with $\gamma = 0.02$ and $\gamma' = 0.2$		
Number of blocks	13	
Cut ratio	$\frac{72978}{284479} \approx 0.2565$	
Dependency ratio	$\frac{39}{78} \approx 0.5$	
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.1696	Figure 4.24: $\mathrm{BC}(H_{B1}, P)$ where P is the partition of the VTB'-DMC algorithm.

Table 4.18: The FM algorithm and the VTB'-DMC approach with respect to the H_{B1} hypergraph.

We see that the results of the IC-hypergraphs that are similar to the results of H_{A1}. So, the FM algorithm leads to a better cut ratio and the VTB'-DMC algorithm leads to a better dependency ratio. Table ?? collects all results.

	H_{A1}	H_{A2}	H_{A3}	H_{A4}
FM with $\gamma = 0.004, FMrest = 10, FMrep = 10$				
Blocks	32	130	110	127
Cut ratio	$\frac{28984}{221142} \approx 0.1311$	$\frac{37288}{266009} \approx 0.1340$	$\frac{60622}{466758} \approx 0.1299$	$\frac{67149}{515951} \approx 0.1301$
Dep. ratio	$\frac{441}{496} \approx 0.8891$	$\frac{3158}{5995} \approx 0.5268$	$\frac{3723}{5995} \approx 0.6210$	$\frac{5369}{8001} \approx 0.6710$
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.0979	≈ 0.3974	≈ 0.3204	≈ 0.0696
Req. time	9 min.	15 min.	31 min.	32 min.
VTB'-DMC with $\gamma = 0.02$ and $\gamma' = 0.2$				
Blocks	32	130	110	111
Cut ratio	$\frac{51897}{221142} \approx 0.2347$	$\frac{66846}{266009} \approx 0.2513$	$\frac{108359}{466758} \approx 0.2322$	$\frac{100522}{515951} \approx 0.1948$
Dep. ratio	$\frac{146}{496} \approx 0.2944$	$\frac{636}{8385} \approx 0.0758$	$\frac{449}{5995} \approx 0.0749$	$\frac{470}{6105} \approx 0.0770$
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.0952	≈ 0.1568	≈ 0.1503	≈ 0.0558
Req. time	4 min.	57 min.	22 min.	22 min.

	H_{B1}	H_{B2}	H_{B3}
FM with $\gamma = 0.004, FMrest = 10, FMrep = 10$			
Blocks	16	255	167
Cut ratio	$\frac{49848}{284479} \approx 0.1752$	$\frac{83521}{577235} \approx 0.1447$	$\frac{175551}{1123170} \approx 0.1563$
Dep. ratio	$\frac{120}{120} = 1$	$\frac{27569}{32385} \approx 0.8513$	$\frac{6915}{13861} \approx 0.4989$
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.0684	≈ 0.0087	≈ 0.0919
Req. time	15 min.	76 min.	122 min.
VTB'-DMC with $\gamma = 0.02$ and $\gamma' = 0.2$			
Blocks	13	292	275
Cut ratio	$\frac{72978}{284479} \approx 0.2565$	$\frac{90096}{577235} \approx 0.1561$	$\frac{126182}{1123170} \approx 0.1123$
Dep. ratio	$\frac{39}{78} \approx 0.5$	$\frac{3712}{42486} \approx 0.0874$	$\frac{601}{42486} \approx 0.0141$
$\frac{\max_{B \in P} \mu(B)}{A}$	≈ 0.1696	≈ 0.0589	≈ 0.3702
Req. time	16 min.	61 min.	137 min.

Table 4.19: Comparison of the approaches FM, GR-Imp and the VTB'-DM.

5 Conclusion and further research perspectives

This thesis deals with the hierarchical hypergraph partitioning of large IC-hypergraphs. For that, the writer introduces IC-hypergraphs and the class of IC-hypergraphs $\mathcal{IC}_{\varepsilon,\varepsilon'}$ where a typical circuit is in $\mathcal{IC}_{0,0}$. Further, the writer introduces some reasons why it is useful to respect the block connectivity graph for the partitioning of an IC-hypergraph. This leads to the definition of the dependencies of a partitioned hypergraph and further to the objective function $(??)$. The minimization of this objective function will be handled by the VTB'-DM or the VTB'-DMC approach where the writer does not know any paper where this approaches were introduced before. For the minimization of the dependencies, the approaches move the partitioning from the hypergraph to its corresponded block connectivity graph. The results of this approaches were compared with the results of a common used algorithm of the topic of the cut minimization, the FM algorithm. These algorithms could only be applied to large IC-hypergraphs if they were coarsened; otherwise these algorithms is too time-consuming. For this, the writer introduces the coarsening approach BMHEC that is a generalization of the coarsening approach MHEC of Karypis et al., where he does not know any paper where this was introduced before. Further, another approach that coarsens a hypergraph by random walks was presented but the results of this approach show that it is unfavourable for the hypergraph coarsening of large IC-hypergraphs. The writer embeds the hierarchical partitioning by itself into the theory of categories. For that, he introduces the category $\mathbf{Hir}$ that describes, for example, the principle of the hierarchical partitioning, and for that he introduces series of arrows of a category, and again, the writer do not know any paper that was introduced before. Other topics use this principle, too. The writer shows this by an example of a multigrid method that is used to solve a linear equation. Further, the writer introduces the $\mathbf{SA}$ category that uses the connection between random walks and the simulated annealing approach and the interpretation of a random series of arrows of a category as a random walk with respect to some restrictions. This category leads to the fact that the used hierarchical partitioning could be interpreted as a simulated annealing. Furthermore, the use of categories leads to the possibility to research the connections between different hierarchical methods, like the hierarchical partitioning and the multigrid method to solve a linear equation, and so on. Also, the category $\mathbf{Hir}$ provides an opportunity to create a summary of the methods that exist ordered by the type of arrow that they belong to. This means, for example, a summary of the approaches of the hierarchical partitioning of an IC-hypergraph ordered by the used coarsening methods, the used partitioning and improve methods, and the used refinement methods. So the combination of the coarse, partition, improve and refinement methods can be analysed and may lead to new combinations or approaches. The same could

be done for other topics that belong to the category **Hir** as the multigrid method, and so on. For example, the results of the VTB'-DM or of the VTB'-DMC approach lead to the fact that we can get better results if we have a partition method that minimizes the cut between several blocks so that the dependency ratio is not too high and then minimizes the dependencies with the VTB'-DM or VTB'-DMC approach. The writer shows in Lemma **??** that the dependency ratio has to respect to minimize this ratio. Maybe this partition method could be achieved by a better initialisation of the blocks in the FM algorithm. Further, the results of the VTB'-DMC approach show that the average number of dependencies could be considered, too. So the VTB'-DMC or the VTB'-DM approach could be modified so that blocks with a lot of dependencies could be split into two blocks, and so on. Also, this results lead to create an improve method that considers the dependency ratio, and so on. The writer is excited to see which further researches will improve the IC design. So he finishes his statements with a funny quote of Thomas Watson:

> *I think there is a world market for maybe five computers.*[1]

[1] See: [**?**]

A Appendix - PCs

Table ?? present two PCs that were used for the calculations.

Name	CPU	RAM	bit
PC1	4×2.0 GHz	16 GB	64
PC2	6×3.06 GHz	48 GB	64

Table A.1: Data of the PCs that were used for the calculation.

Bibliography

[1] *ISPD 2006 ACM International Symposium on Physical Design April 9-12 2006, San Jose CA*. http://archive.sigda.org/ispd2006/contest.html.

[2] *Things People Said Bad Predictions*. http://www.rinkworks.com/said/predictions.shtml.

[3] Awodey, S.: *Category Theory*. Oxford University Press, 2 edition, 2010.

[4] Berge, C.: *Hypergraphs*. North-Holland mathematical libary, 1989.

[5] Berge, C. Ray-Chaudhuri, D.: *Hypergraph Seminar*. Springer-Verlag, 1972.

[6] Bollobás, B.: *Modern Graoh Theory*. Springer, 1998.

[7] Cooper, C. Frieze, A. Radzik T.: *The cover time of random walks on hypergraphs*. http://www.math.cmu.edu/ af1p/Texfiles/eavesdrop.pdf, 2010.

[8] Ding, C.H.Q.; Xiaofeng H.; Hongyuan Z.; Ming G.; Simon, H.D.: *A min-max cut algorithm for graph partitioning and data clustering*. Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on, 1:107 – 114, 2001.

[9] Fiduccia, C.M. ; Mattheyses, R.M.: *A Linear-Time Heuristic for Improving Network Partitions*. Design Automation, 19:175 – 181, 1982.

[10] Garey, M.; Johnson, D.: *COMPUTERS AND INTERACTABILITY A Guide to the Theory of NP-Completeness*. W. H. FREEMAN AND COMPANY, 1979.

[11] Gong, J.; Sung K. L.: *Multiway partitioning with pairwise movement*. Computer-Aided Design, 1998. ICCAD 98. Digest of Technical Papers. 1998 IEEE/ACM International Conference on, 1:512 – 516, 1998.

[12] Heinicke, F.: *Untersuchung von heuristischen Optimierungsverfahren zur hierarchischen Platzierung von Standardzellen in 3D-Aufbauten*. Master's thesis, Hochschule Mittweida, 2010.

[13] Karypis, G.; Aggarwal, R. ; Kumar V. ; Shekhar S.: *Multilevel Hypergraph Partitioning: Applications in VLSI Domain*. IEEE, 7(1):69–79, 1999.

[14] Kaufmann, M.; Wagner, D.: *Drawing Graphs: Methods and Models*. Springer, 2001.

[15] KLENKE, A.: *Wahrscheinlichkeitstheorie*, volume 2. Springer-Verlag, 2008.

[16] KRISHNAMURTHY, B.: *An Improved Min-Cut Algonthm for Partitioning VLSI Networks*. Computers, IEEE Transactions on, C-33, Issue: 5:438 – 446, 1984.

[17] MITZENMACHER, M.; UPFAL, E.: *Probability and Computing Randomized Algortihms and Probablistic Analysis*. Cambrig University Press, 2005.

[18] POINCARÉ, HENRI: *Wissenschaft und Hypothese*, volume 4. Xenomoi, 2003.

[19] SANCHIS, L.A.: *Multiple-way network partitioning*. Computers, IEEE Transactions on, 38, Issue: 1:62 – 81, 1989.

[20] SCHNITGER, PROF. DR. GEORG: *Effiziente Algorithmen*. http://www.thi.informatik.uni-frankfurt.de/Effiziente10/kapitel3h.pdf.

[21] SCHWETLICK, H.; KRETZSCHMAR, H.: *Numerische Verfahren für Naturwissenschaftler und Ingenieure*. Fachbuchverlag Leipzig, 1991.

[22] SCHWOPE, A. http://www.andreas-schwope.de/ASIC_s/Entwicklung/Beispiel/beispiel.html.

[23] TITTMANN, P.: *Graphentheorie*. Carl Hanser Verlag, 2003.

[24] WOESS, W.: *Random Walks on Infinite Graphs and Grouos*. Cambrig University Press, 2000.

[25] WONDROUS, STRANGE. http://strangewondrous.net/browse/author/p/poincare+henri?start=11.

[26] ZHOUY, D.; HUANGZ, J.; SCHÖLKOPF B.: *Learning with Hypergraphs: Clustering,Classification, and Embedding*. Advances in Neural Information Processing Systems (NIPS) 19, 2007.

Declaration of Academic Honesty

Hereby I, Alexander Steinhardt, assure to have written the present thesis with the title "Consideration of local structures in hierarchical partitioning of integrated circuit netlists modelled by hypergraphs" on my own and that I marked all the literature and resources used.

Mittweida, 27. February 2013

Alexander Steinhardt